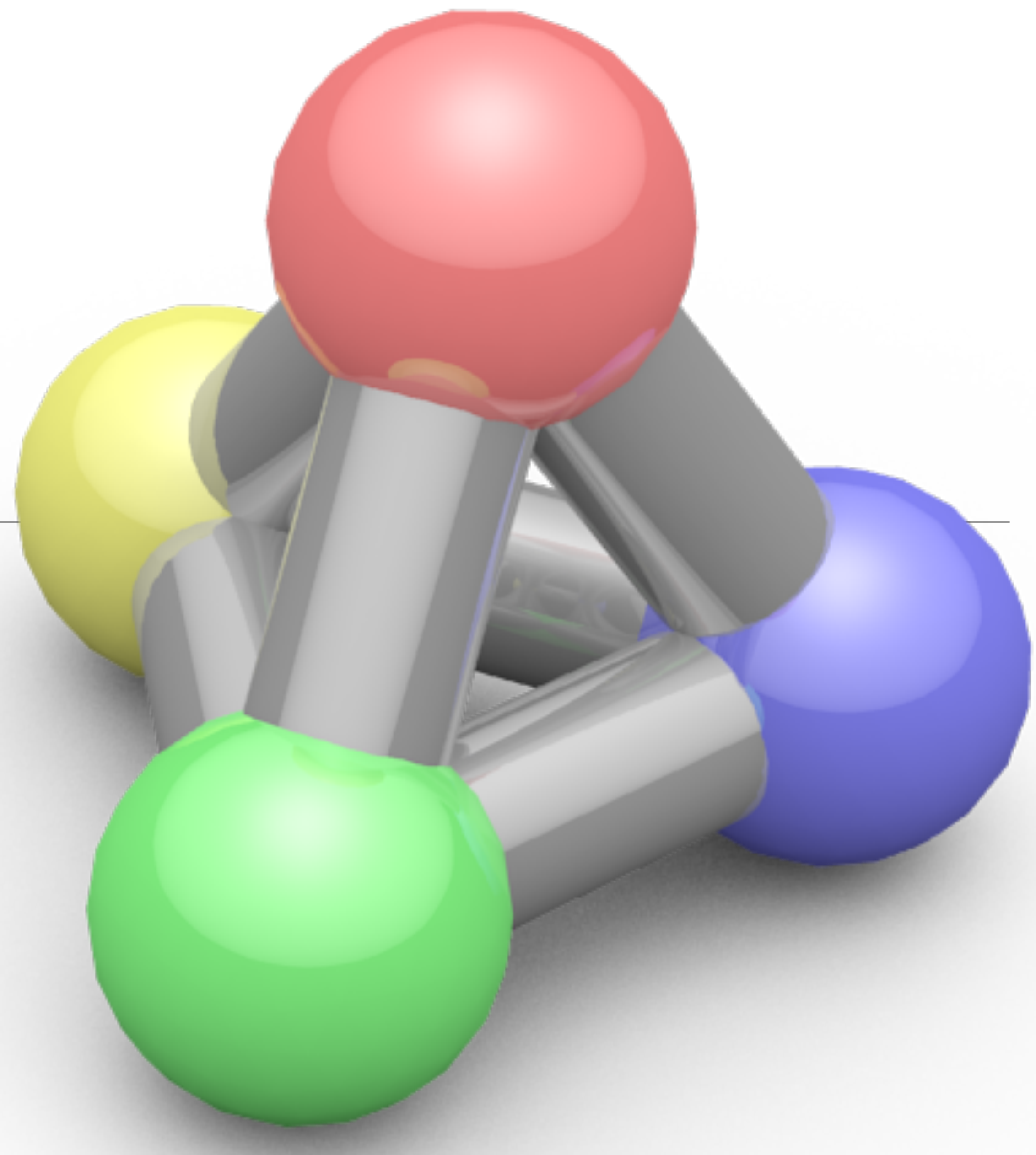
# Computer Animation



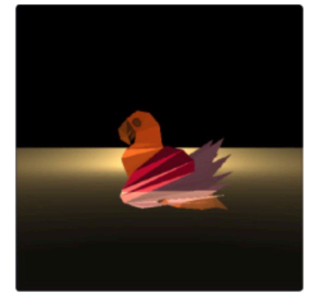
CPSC 453 – Fall 2016

Sonny Chan

# Reminder:
# Remember to **vote!**

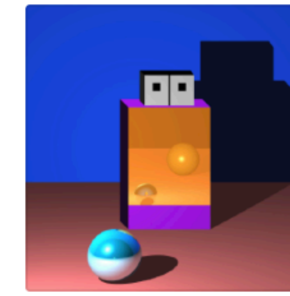
---

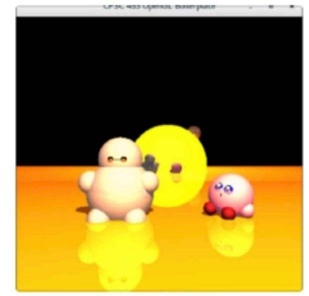
Polls close on Wednesday


Turkey


Teddy Bear

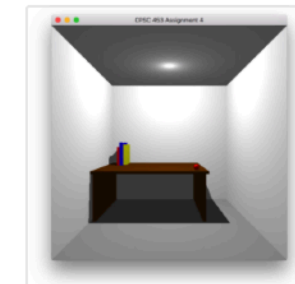

Night


USB

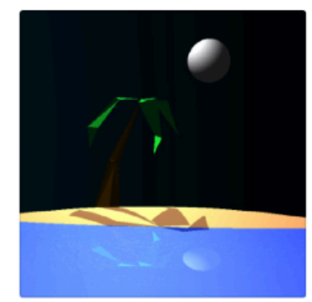

Baymax & Kirby

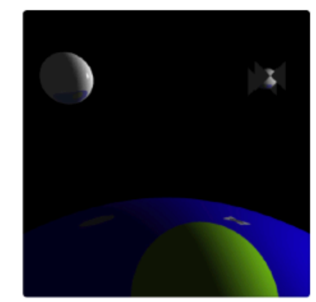

Midas


Twiggy Snowman


Desk in Room

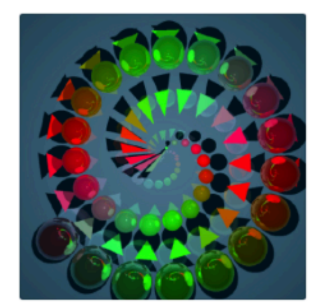

Snowman with Hat


Tropical Island

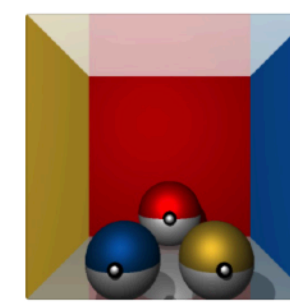

Orbit

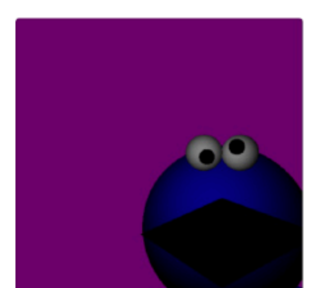

Rover & Trees

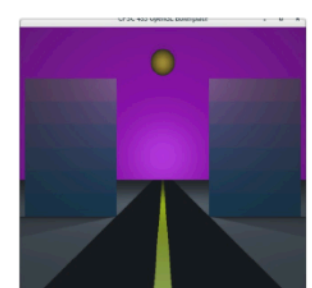

Saruman atop Orthanc

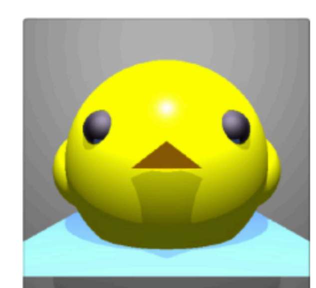

Spiral

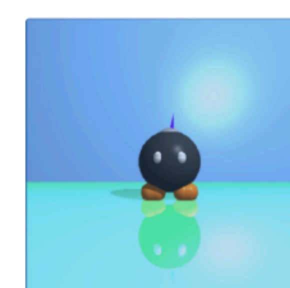

Christmas Tree


Pokeballs


Cookie Monster


Sunset Road


Yellow Bird


Bomberman

Animation is the act, process, or result of imparting life, interest, spirit, motion, or activity.

**–Michael Ashikhmin (Chapter 16)**

# Outline for Today

**Principles**

**Keyframes**

**Procedural**

**Physics-based**

**Characters**

**Motion capture**

# Principles of Animation

Luxo Jr. – Pixar Animation

“There is no particular mystery in animation… it’s really very simple, and like anything that is simple, it is about the hardest thing in the world to do.”

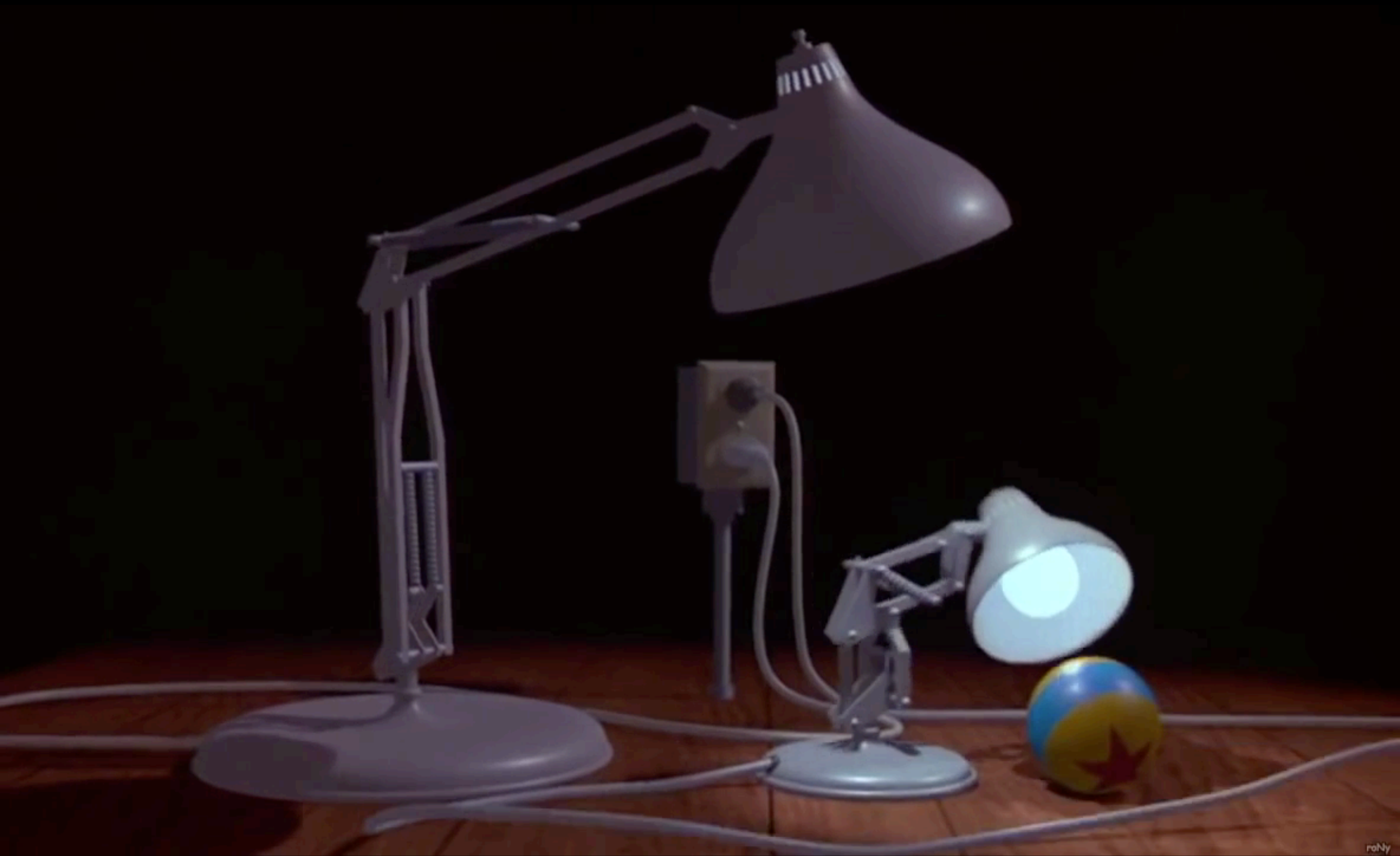
**–Bill Tytla, Walt Disney Studio, 1937**

The Illusion of Life

Cento Lodigiani

“The thing I wanted to do in Luxo Jr. was make the characters and story the most important thing, not the fact that it was done with computer graphics.”

**–John Lasseter, Pixar Animation, 2013**



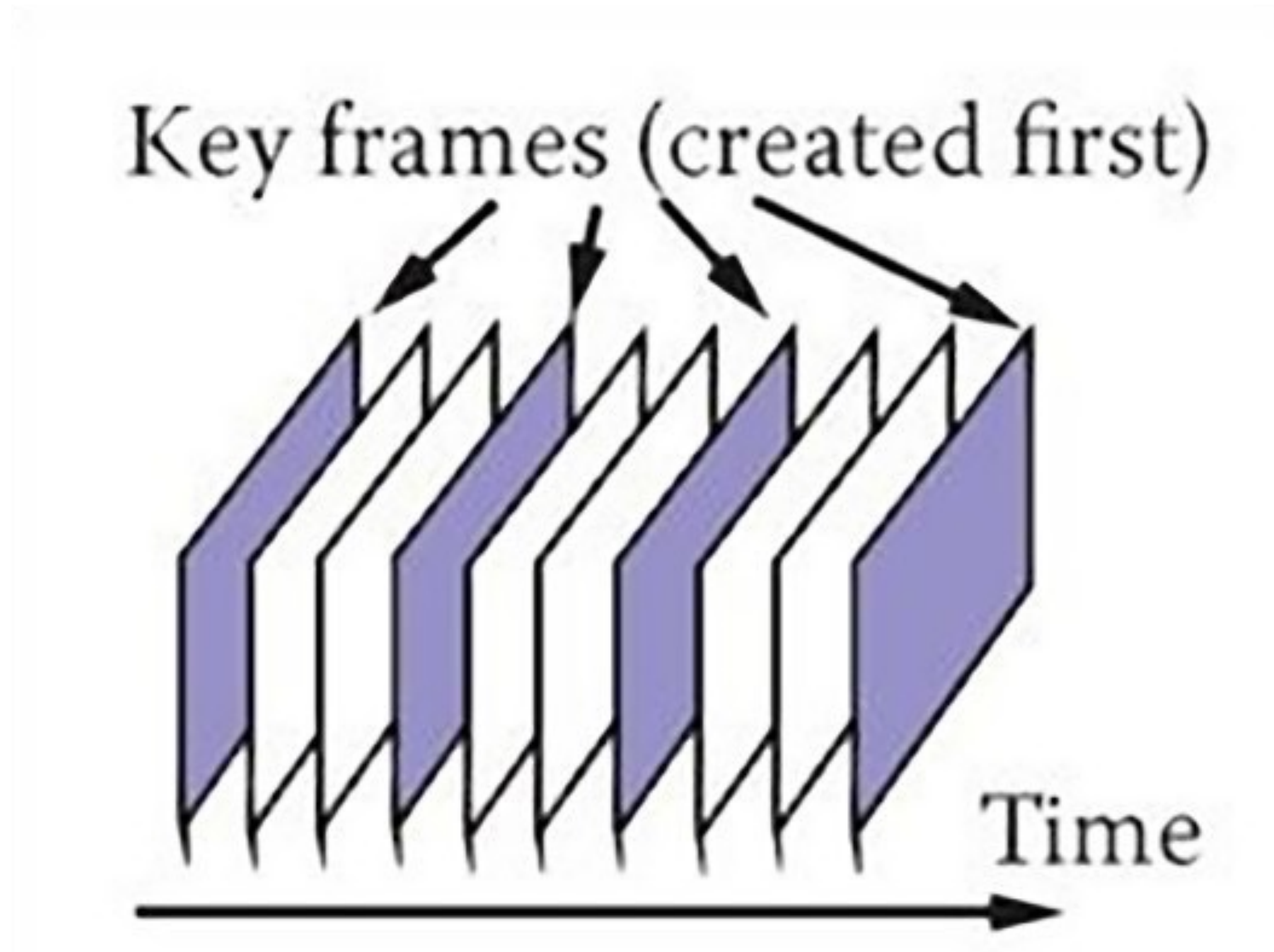
Luxo Jr., Pixar Animation Studios, 1986

# Twelve principles: which ones did you notice?

- squash and stretch
- timing
- anticipation
- follow through and overlapping action
- slow-in and slow-out
- staging
- arcs
- secondary action
- straight-ahead and pose-to-pose action
- exaggeration
- solid drawing skill
- appeal

# Keyframe Animation

# Key Frames



# What properties

can we animate using key frames?

Keyed Properties

**Position**

**Angle / Orientation**

**Shape / Deformation**

**Colour / Material**

**Brightness / Light**

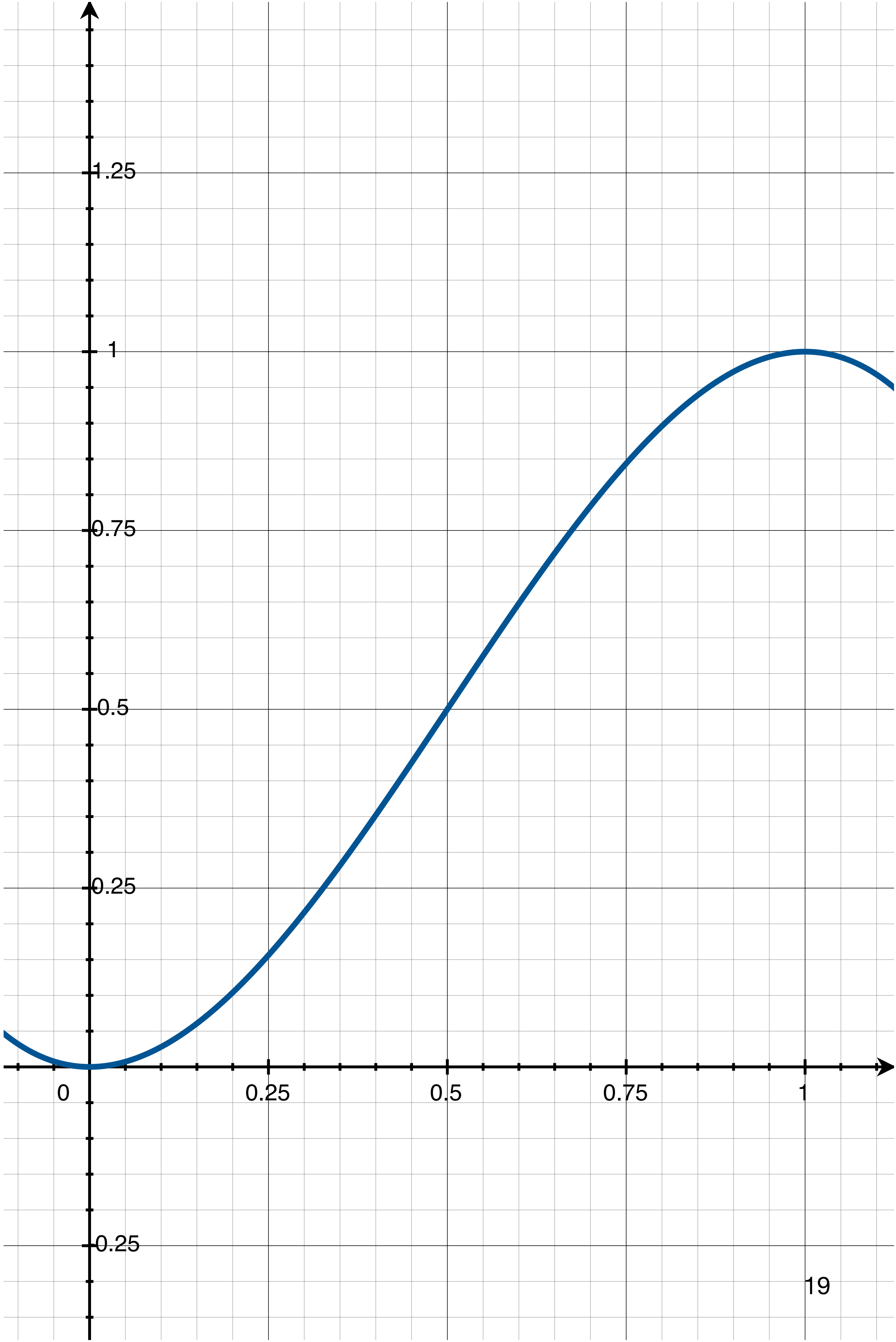
**Visibility**

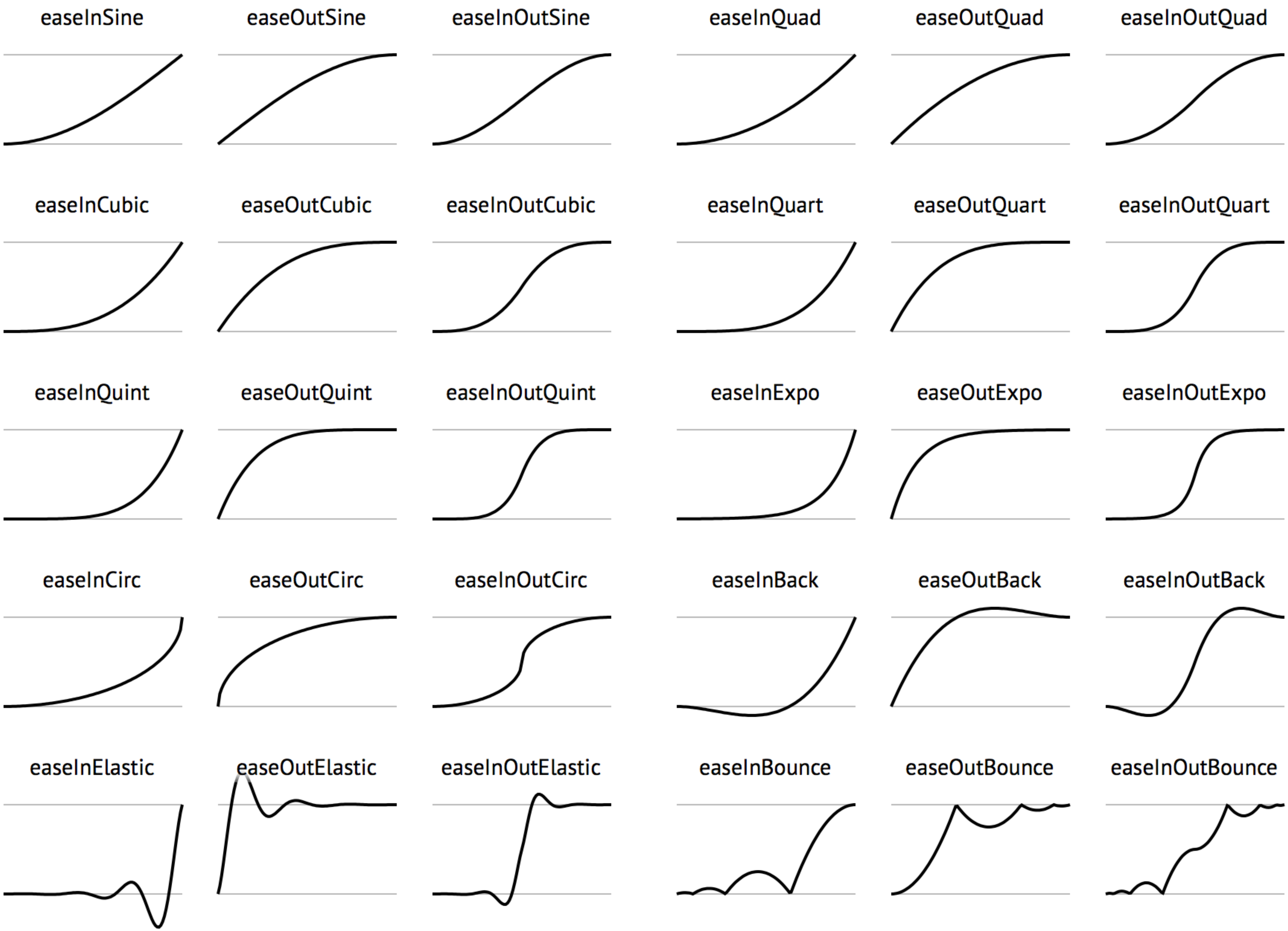
Animation is all about

# timing!

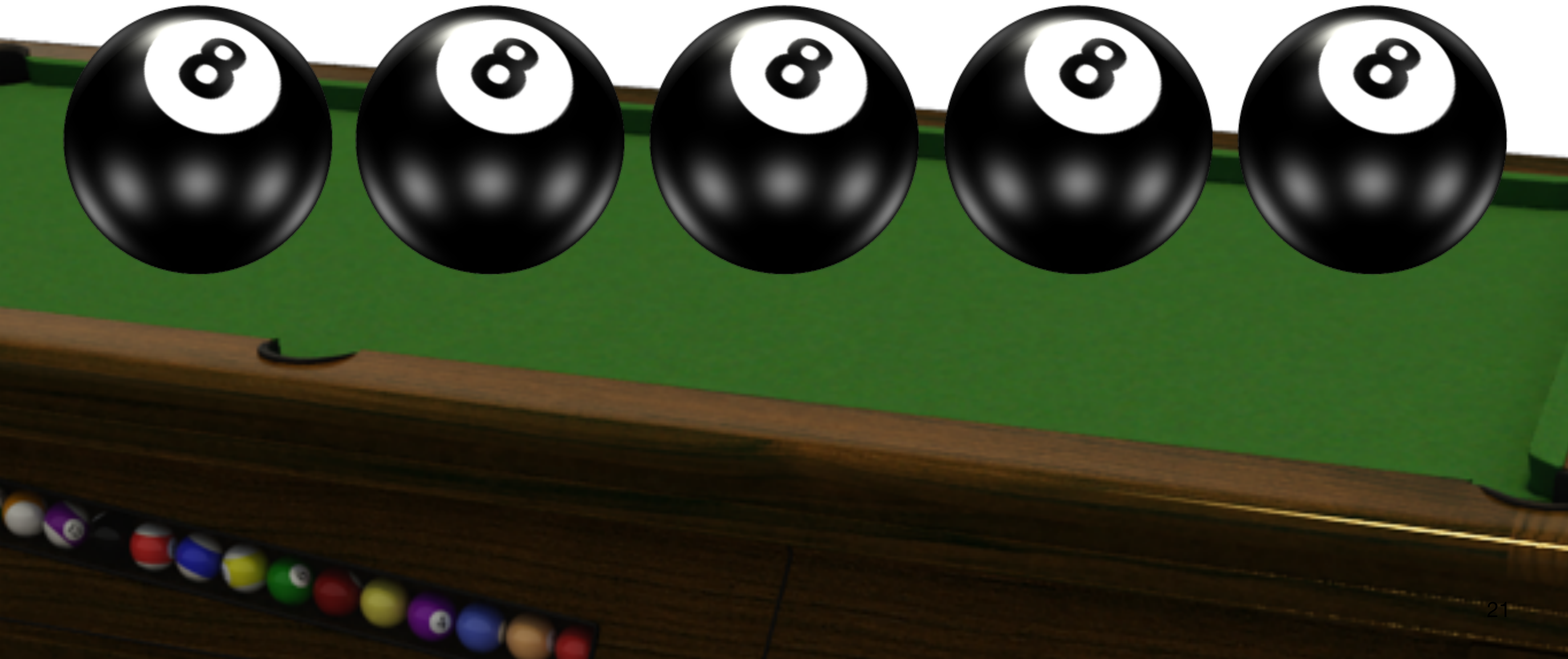
# Cubic Ease Function

$$u(t) = t^2(3 - 2t)$$
$$t \in [0, 1]$$



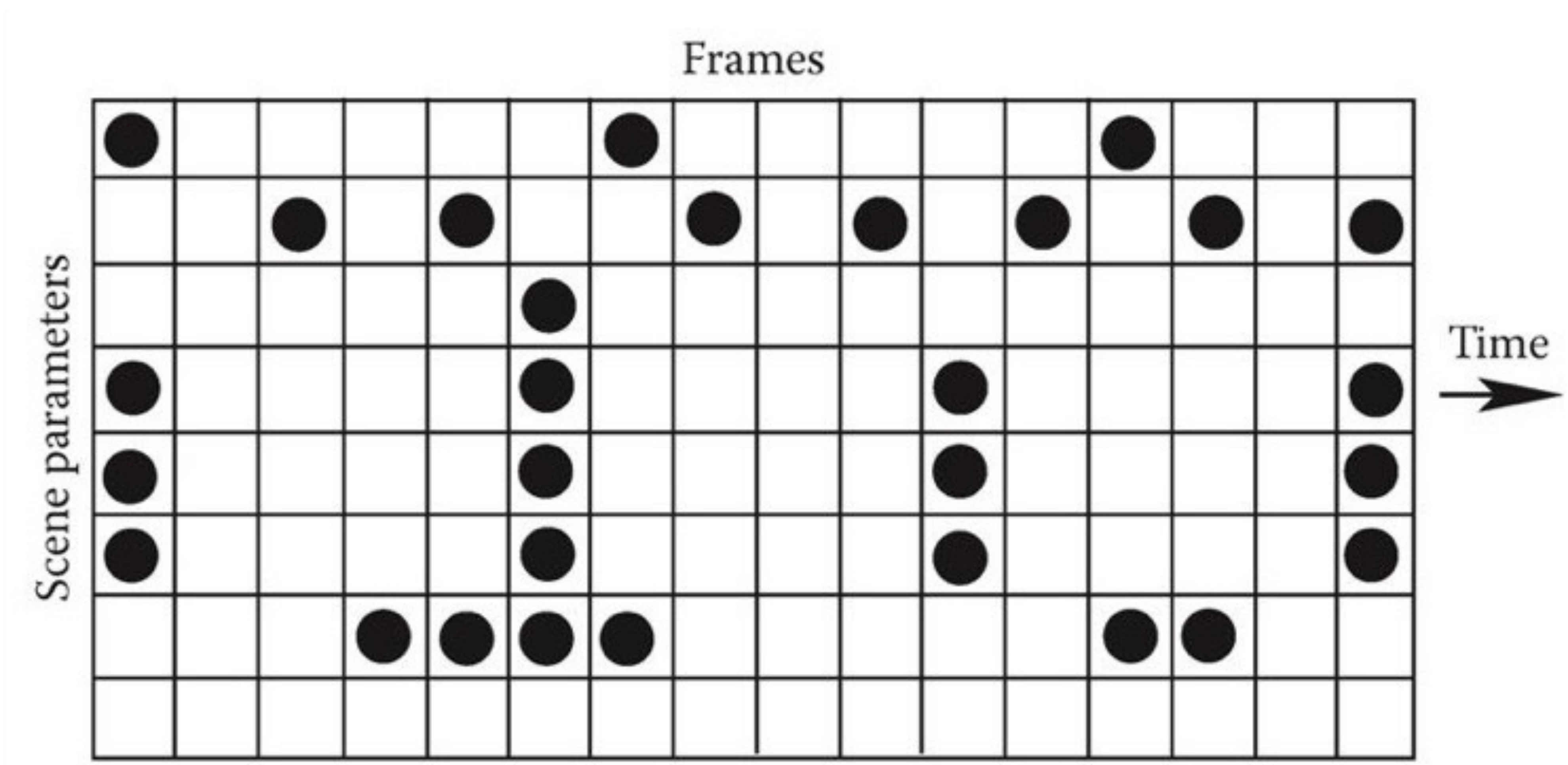



[from easings.net]



# Keyframe Interpolation

- Each animation parameter is keyed at various times
- Ease functions are used to interpolate between values

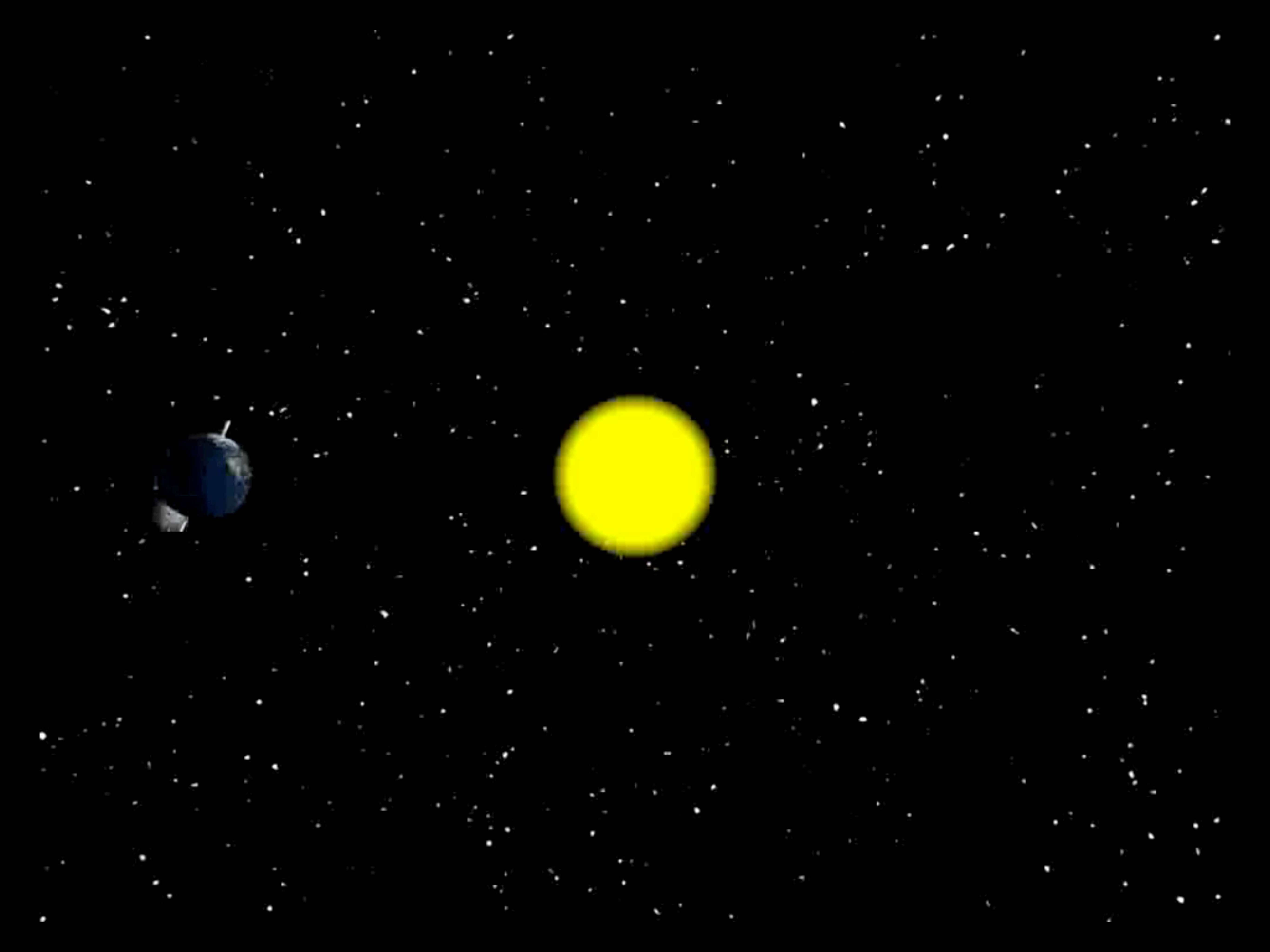


# Procedural Animation

For the special cases when a **mathematical function** outputs precisely the desired motion, given some animator guidance.

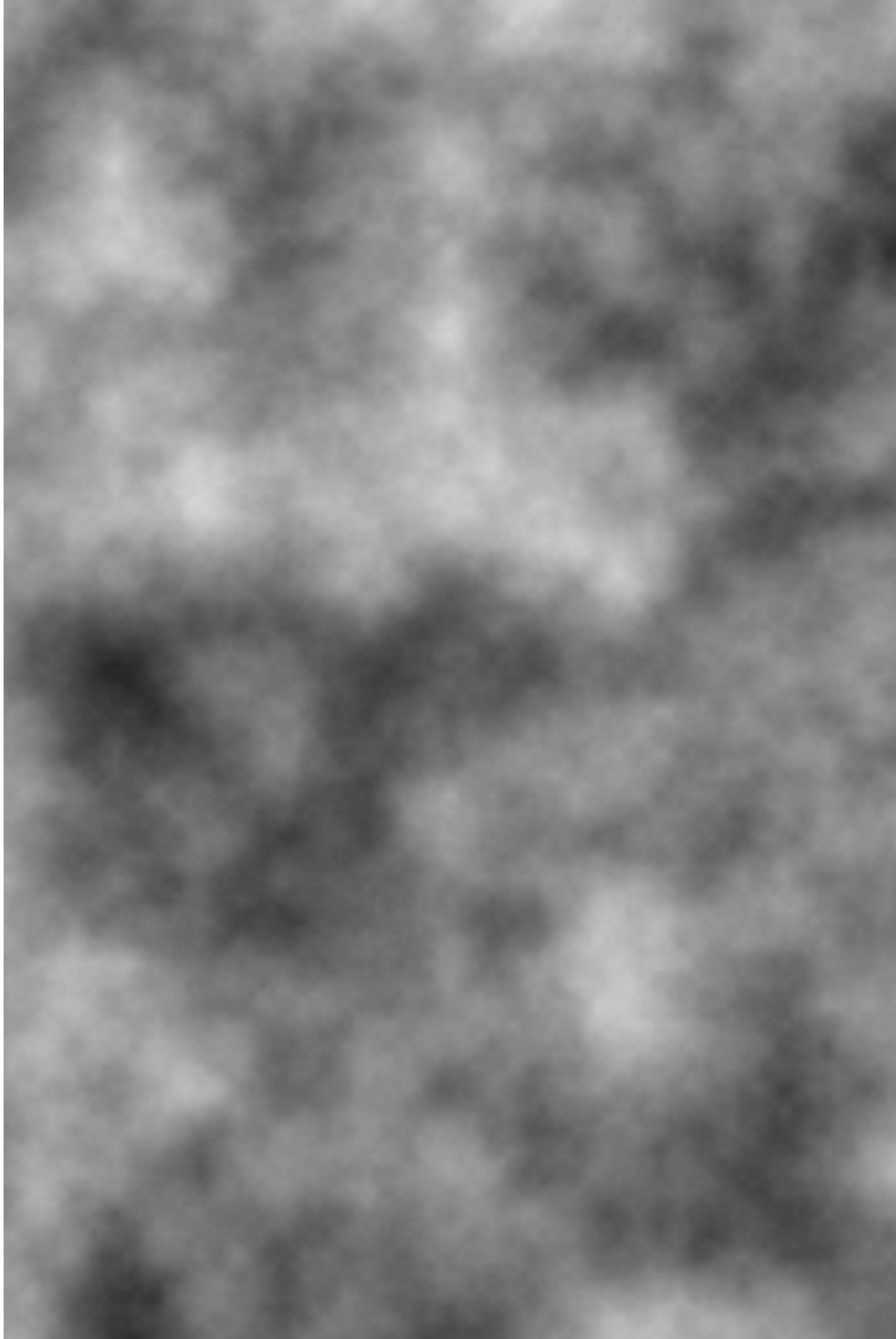
# Planetary Orbits

$$\theta(t) = \omega\, t$$
$$x(t) = r\cos\left(\theta(t)\right)$$
$$y(t) = r\sin\left(\theta(t)\right)$$



# Perlin Fractal Noise

- Define a grid over the domain (1D, 2D, or 3D)
- Generate a random vector at each grid point
- Compute dot products at interior points, interpolate
- Scale and add on to original function to produce fractal noise

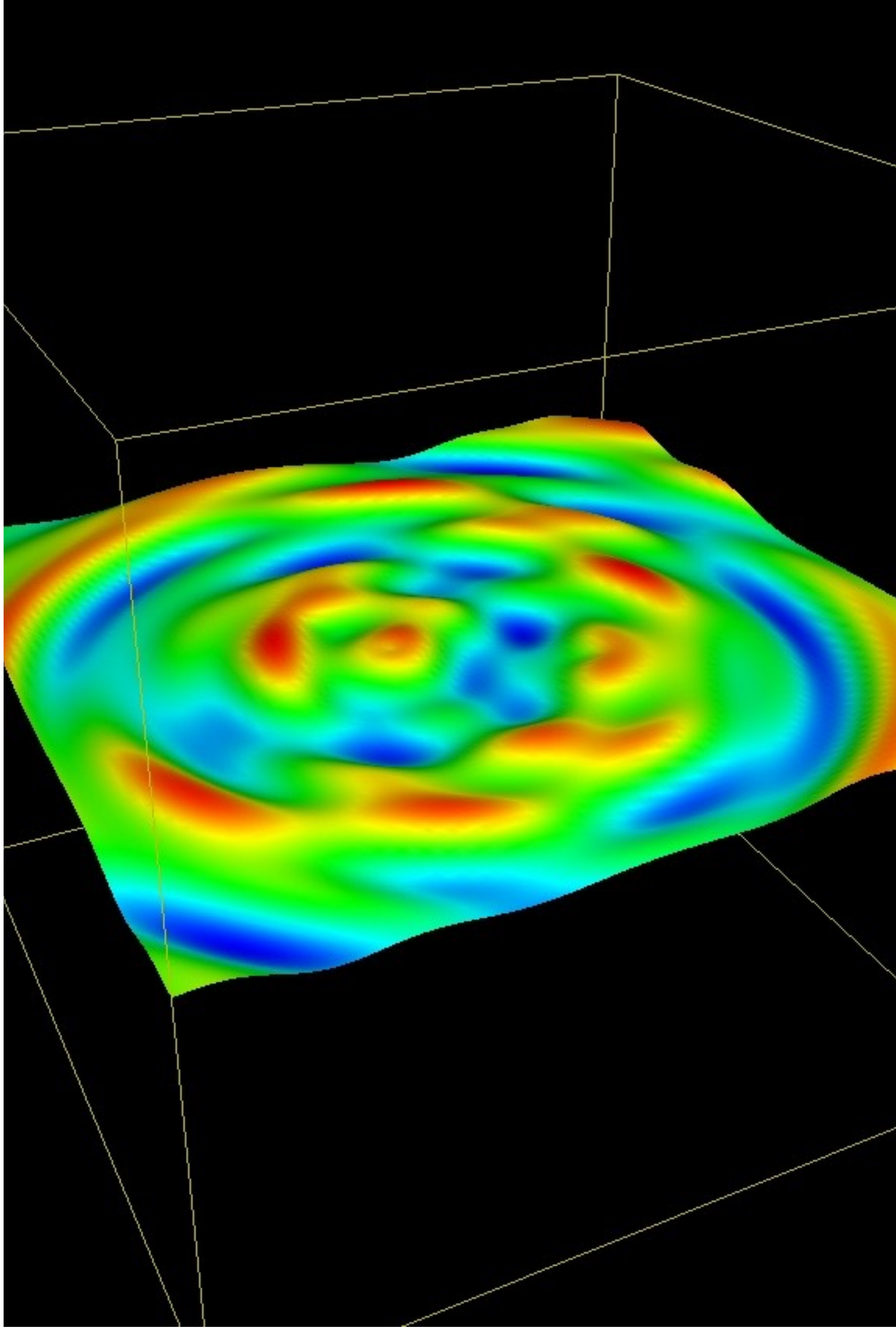


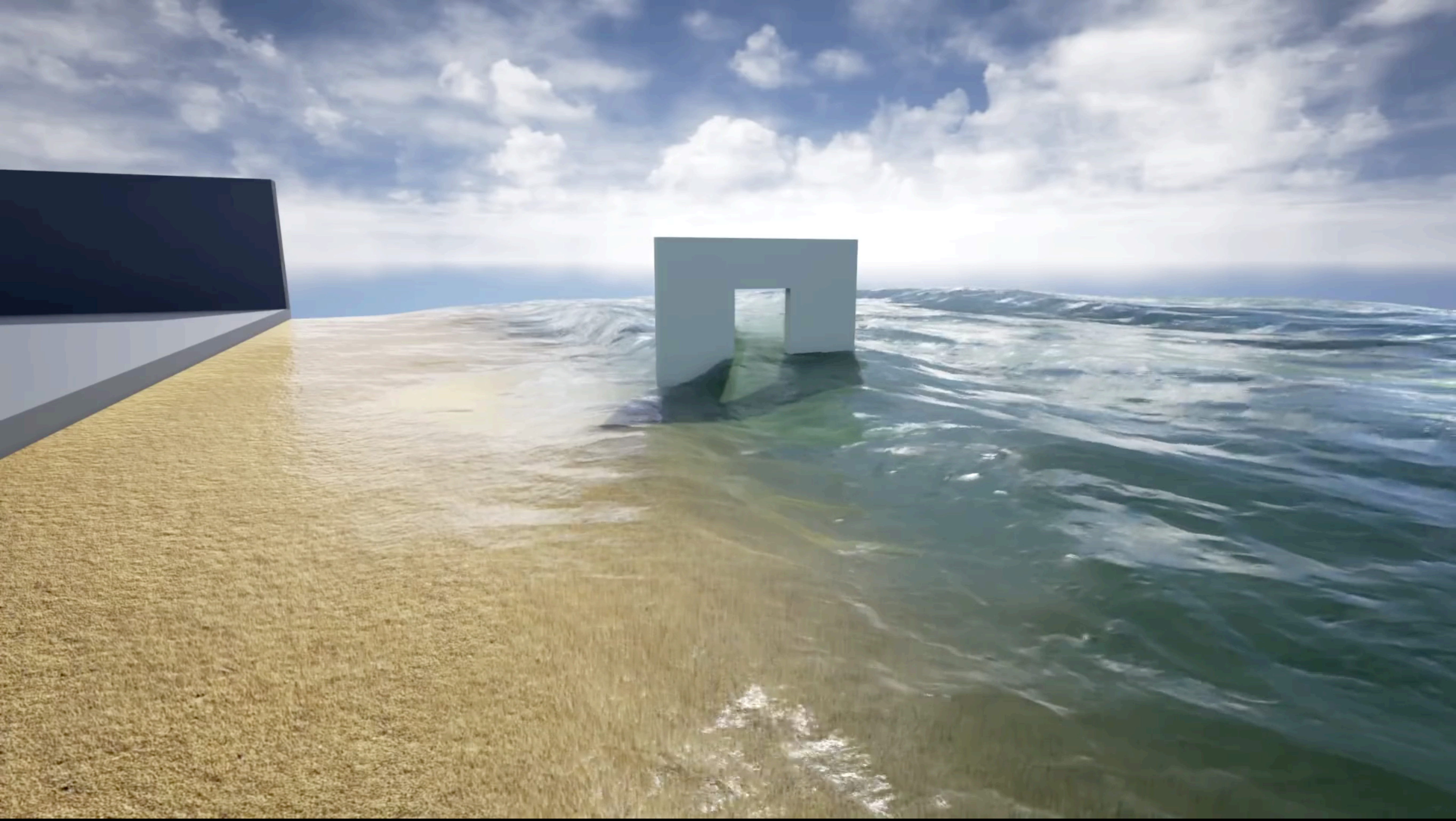
Procedural Clouds | DigitalRune Engine (XNA)

# Sine Waves

$$z = \sum_{i}^{n} A_i \sin\left(f_i\, r_i\, t\right)$$

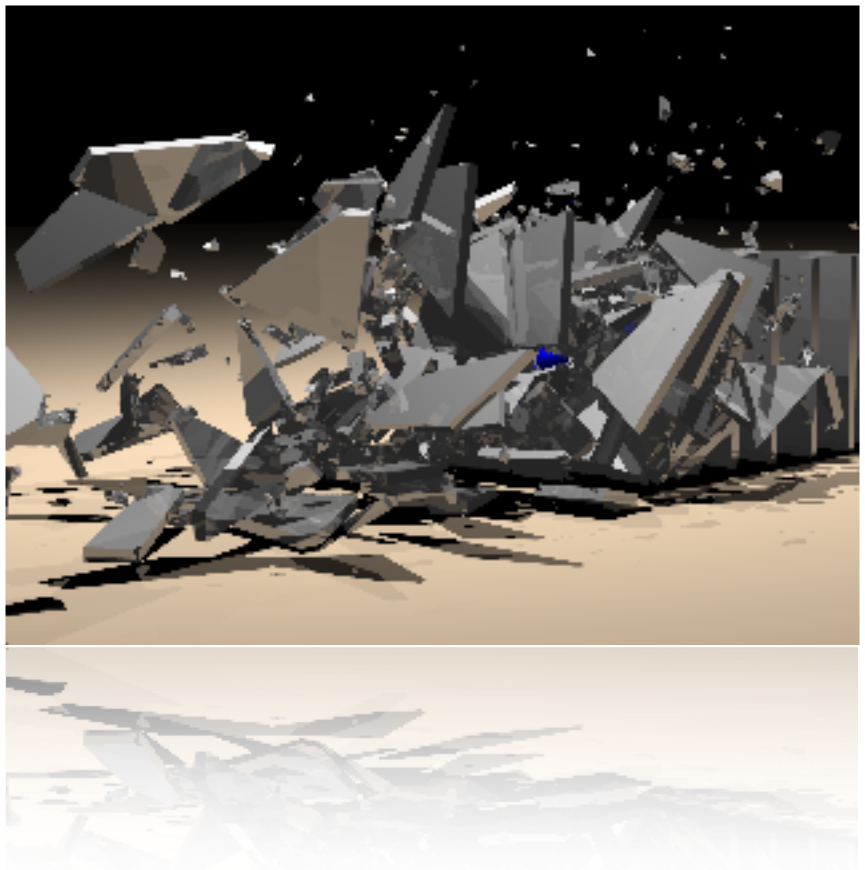
$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$$



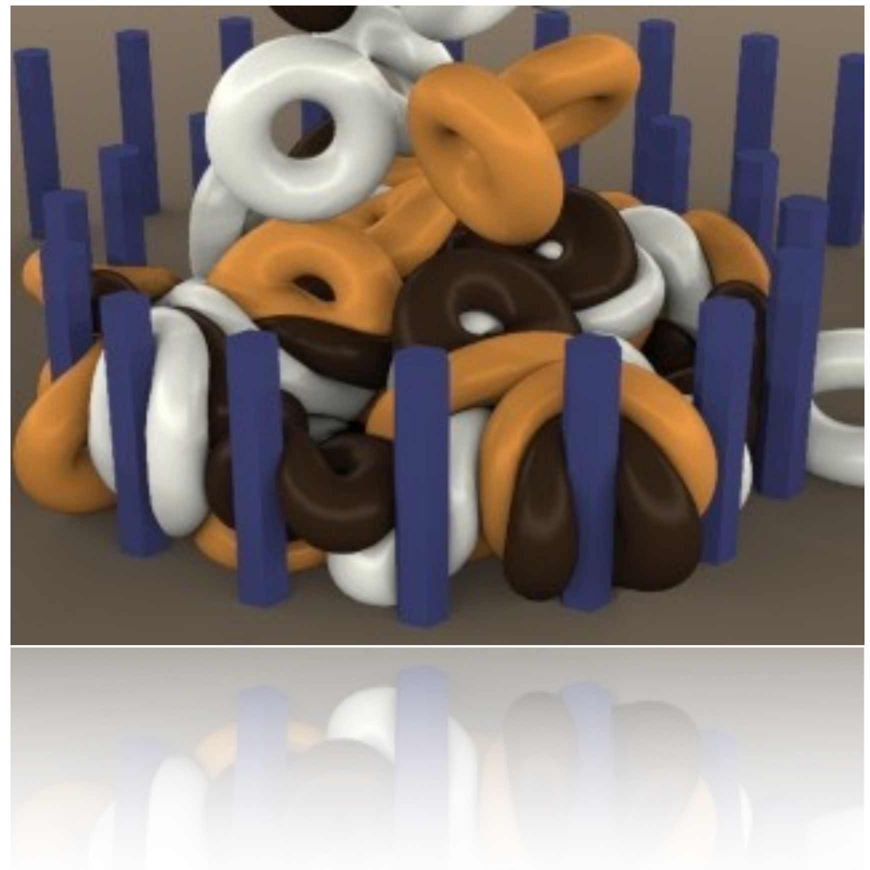


Ocean Waves | Unreal Engine 4, Ben Allen

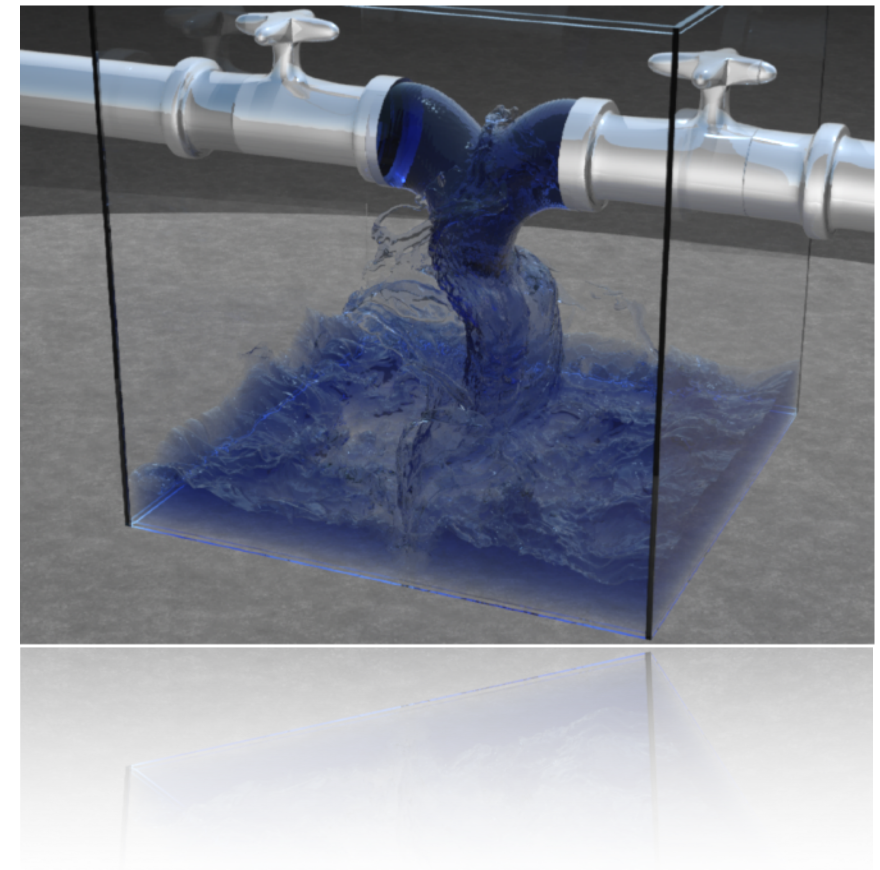
# Physics-Based Animation



Rigid Bodies



Deformable Objects



Fluids

[images from Ron Fedkiw, Stanford University]

Newton's second law of motion:

$$F = ma$$

# Physics Simulation

- Physical quantities are related/defined by differential equations
- Apply forces to objects, solve for displacements
- If we know the force on an object, and its mass, how might we find its position at a given time?

$$\mathbf{F} = m\mathbf{a} \qquad \mathbf{a} = \frac{1}{m}\mathbf{F}$$

$$\mathbf{a} = \frac{d}{dt}\mathbf{v} \qquad \mathbf{v} = \int \mathbf{a}\, dt$$

$$\mathbf{v} = \frac{d}{dt}\mathbf{r} \qquad \mathbf{r} = \int \mathbf{v}\, dt$$

# Numerical Integration

- Forces are often unknown *a priori*
  - no closed form solution
  - we can apply a discrete time simulation
- Use numerical integration to solve differential equations
  - e.g. first order taylor expansion:

$$\mathbf{v}(t) = \int \mathbf{a}(t)\, dt \quad \Rightarrow \quad \mathbf{v}(t_{i+1}) = \mathbf{v}(t_i) + \Delta t\, \mathbf{a}(t_i)$$

# Mass Particles

Cinema 4D


Multi-Physics

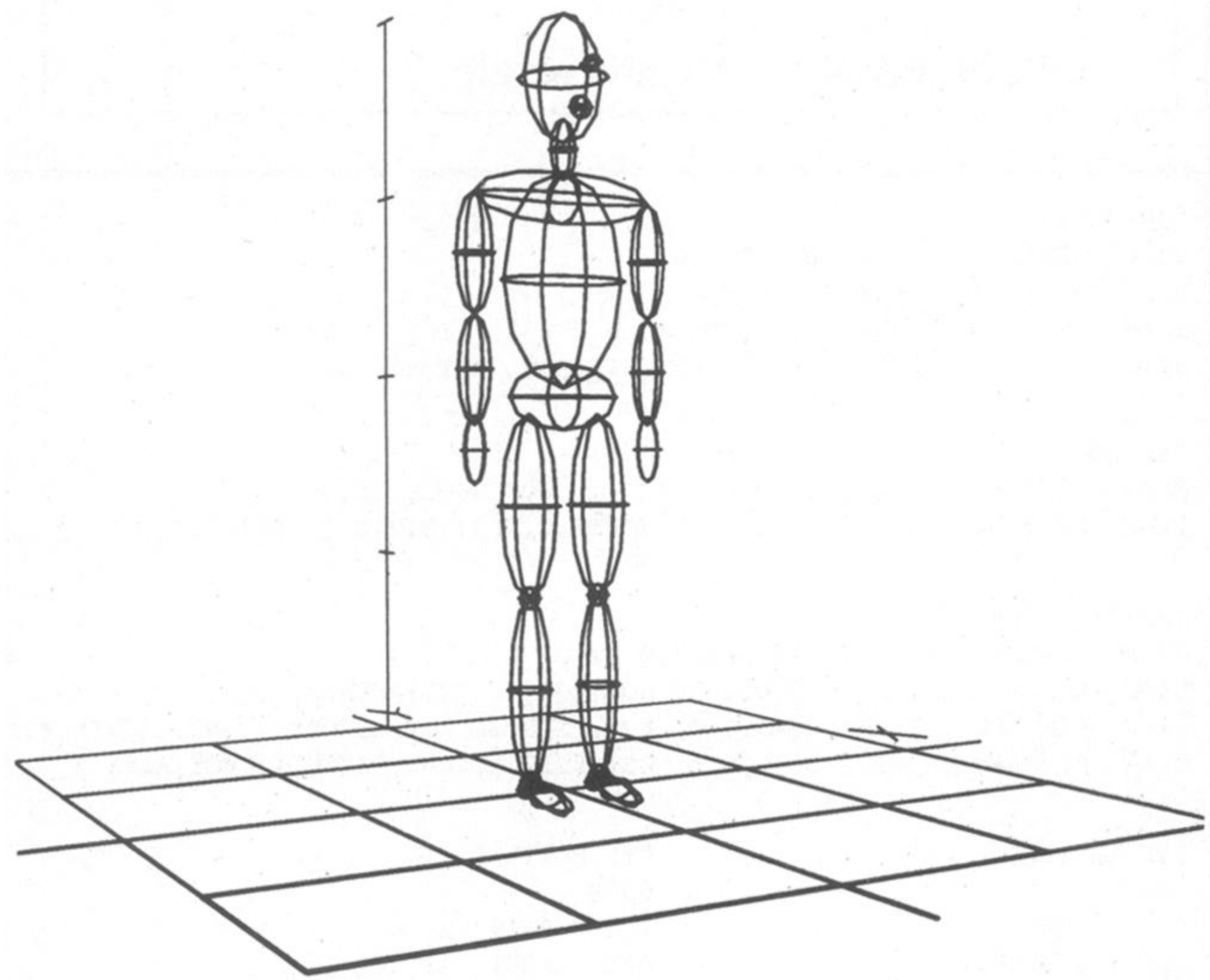
NVIDIA PhysX Flex

# Character Animation

# James F. Blinn

Nested Transformations &
Blobby Man, 1987

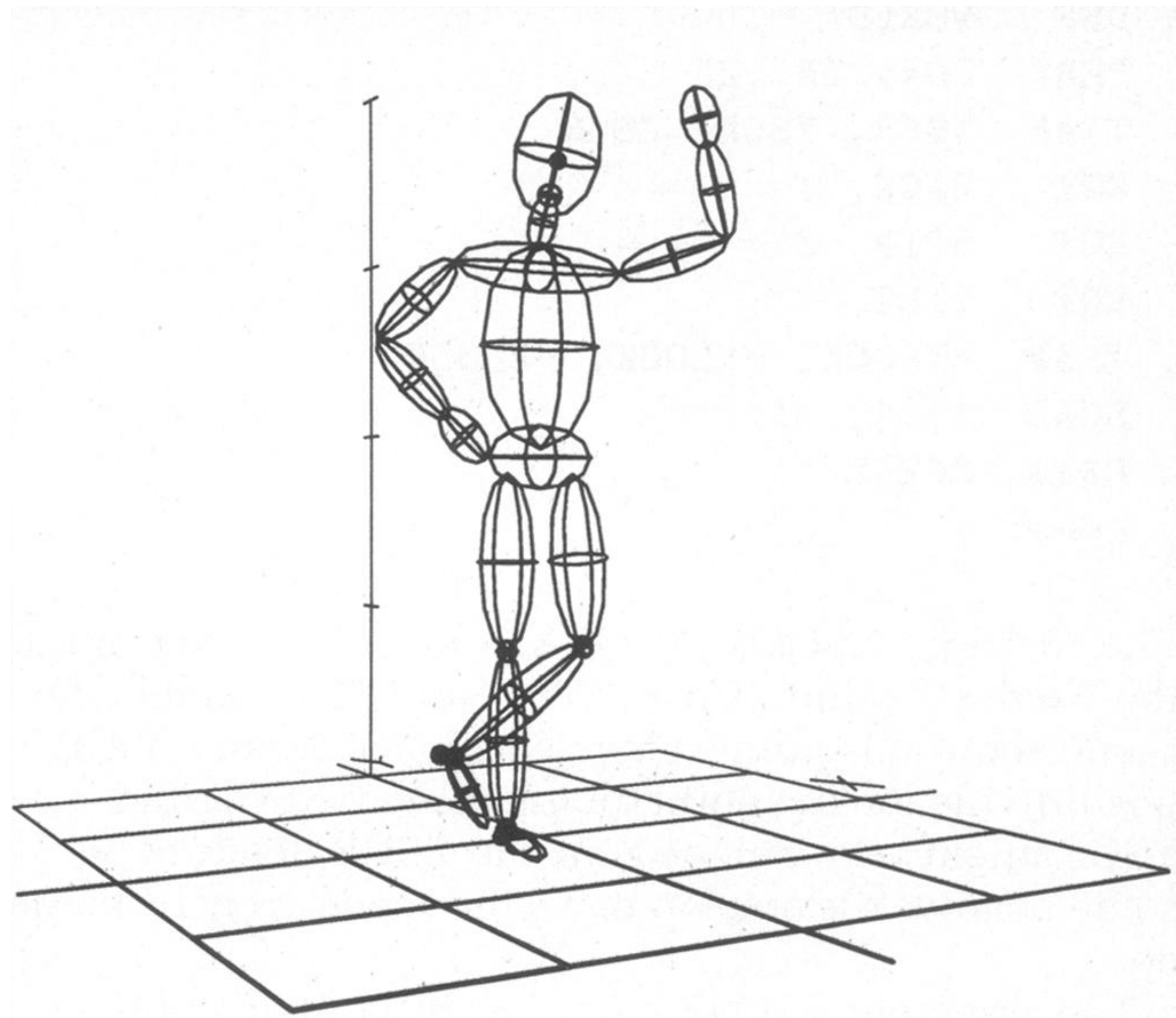
```
DEF TORSO
DRAW  LEFTLEG , TRAN,-0.178,0,0,
DRAW  RGHTLEG , TRAN,0.178,0,0,
DRAW  SPHERE  , TRAN,0,0,0.08, SCAL,0.275,0.152,0.153,
DRAW  BODY    , ROT,EXTEN,1, ROT,BTWIS,2, ROT,ROT,3,
----
DEF BODY
DRAW  SPHERE  , TRAN,0,0,0.62, SCAL,0.306,0.21,0.5,
DRAW  SHOULDER, TRAN,0,0,1, ROT,EXTEN,1, ROT,BTWIS,2, ROT,ROT,3,
----
DEF SHOULDER
DRAW  SPHERE  , SCAL,0.45,0.153,0.12,
DRAW  HEAD    , TRAN,0,0,0.153, ROT,NOD,1, ROT,NECK,3,
DRAW  LEFTARM , TRAN,-0.45,0,0, ROT,LSID,2, ROT,LSHOU,1, ROT,LATWIS,3,
DRAW  RGHTARM , TRAN, 0.45,0,0, ROT,RSID,2, ROT,RSHOU,1, ROT,RATWIS,3,
----
DEF LEFTLEG                      DEF RGHTLEG
PUSH                             PUSH
ROT   LHIP, 3,                   ROT   RHIP, 3,
ROT   LOUT, 2,                   ROT   ROUT, 2,
ROT  -LHIP, 3,                   ROT  -RHIP, 3,
ROT   LTWIS, 3,                  ROT   RTWIS, 3,
DRAW  THIGH   ,                  DRAW  THIGH   ,
TRAN  0, 0, -0.85,               TRAN  0, 0, -0.85,
ROT   LKNEE, 1,                  ROT   RKNEE, 1,
DRAW  CALF    ,                  DRAW  CALF    ,
TRAN  0, 0, -0.84,               TRAN  0, 0, -0.84,
ROT   LANKL, 1                   ROT   RANKL, 1
DRAW  FOOT                       DRAW  FOOT
POP                              POP
----                             ----
DEF LEFTARM                      DEF RGHTARM
PUSH                             PUSH
DRAW  UPARM                      DRAW  UPARM
TRAN  0, 0, -0.55,               TRAN  0, 0, -0.55,
ROT   LELBO, 1,                  ROT   RELBO, 1,
DRAW  LOWARM                     DRAW  LOWARM
TRAN  0, 0, -0.5,                TRAN  0, 0, -0.5,
DRAW  HAND                       DRAW  HAND
POP                              POP
----                             ----
```



[from J. Blinn, *IEEE Computer Graphics & Applications*, 1987]

**Table 1. Meanings of Blobby Man variables**

| | |
|---|---|
| EXTEN | Extension. A dancers term for bending forward and backwards ($x$ axis) |
| ROT | Rotation. A dancers term for rotating the body and shoulders left and right about the vertical ($z$ axis) |
| BTWIS | Angle of body leaning left and right ($y$ axis) |
| NOD | Head nod |
| NECK | Head shake |
| LHIP,RHIP | Angular direction that the leg is kicked |
| LOUT,ROUT | Angular distance that the leg is kicked |
| LTWIS,RTWIS | Angle the leg is twisted about its length |
| LKNEE,RKNEE | Knee bend |
| LANKL,RANKL | Ankle bend |
| LSID,RSID | Arm rotation to side |
| LSHOU,RSHOU | Arm rotation forward and back |
| LATWIS,RATWIS | Arm rotation about its own length |
| LELBO,RELBO | Elbow angle |



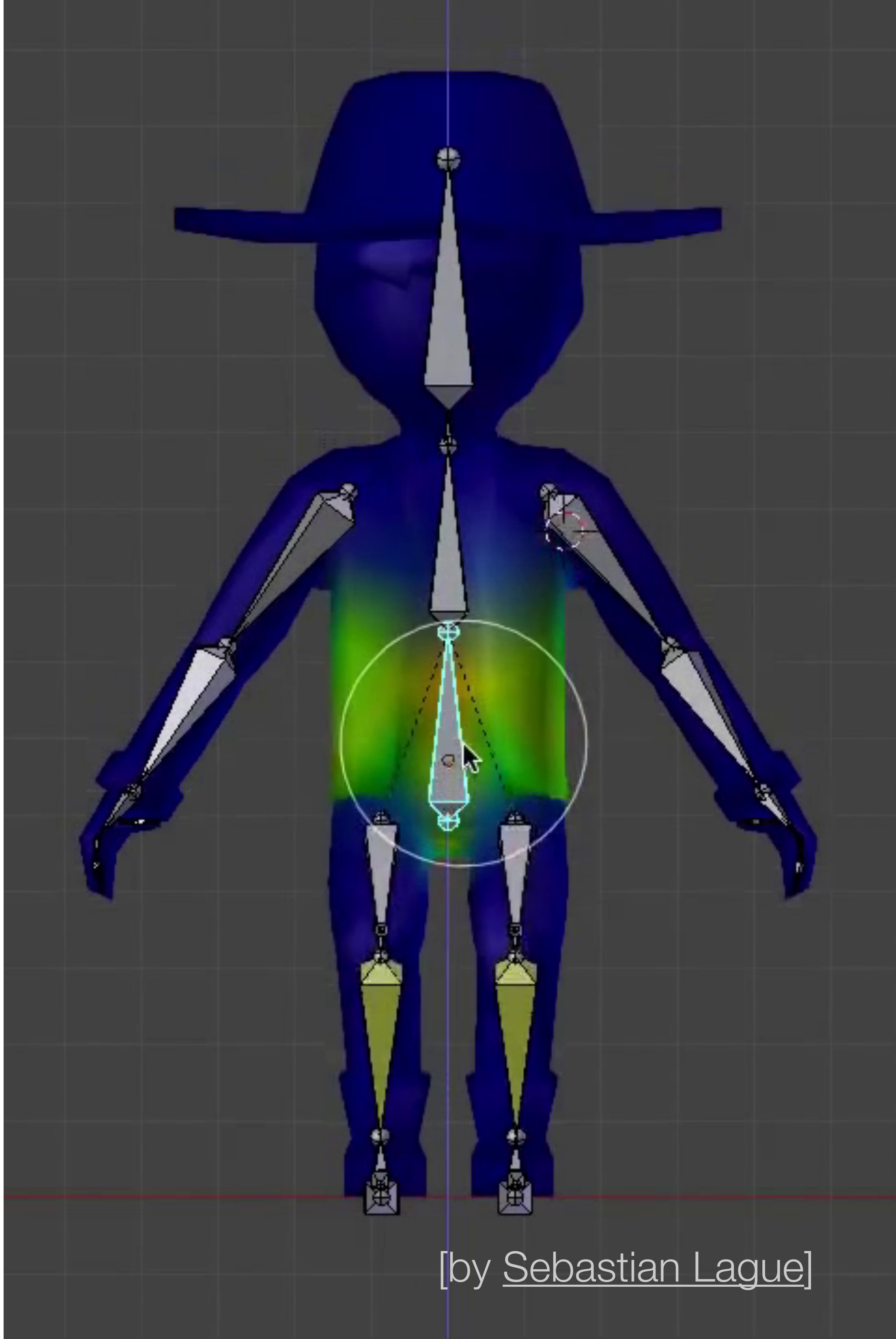
| | | | |
|---|---|---|---|
| NOD=-25 | NECK=28 | | |
| RHIP=105 | ROUT=13 | RTWIS=-86 | RKNEE=-53 |
| LHIP=0 | LOUT=0 | LTWIS=0 | LKNEE=0 |
| LSID=-45 | LSHOU=0 | LATWIS=-90 | LELBO=90 |
| RSID=112 | RSHOU=40 | RATWIS=-102 | RELBO=85 |

[from J. Blinn, *IEEE Computer Graphics & Applications*, 1987]


BOHICA
2 / 5
VS
KLUDGE
0 / 5
NPS: 0.0

# Rigging

- Create a “skeleton” that underlies the object mesh
- Attach the mesh geometry to move with skeleton
- Animate the character by controlling the joint angles of the skeleton



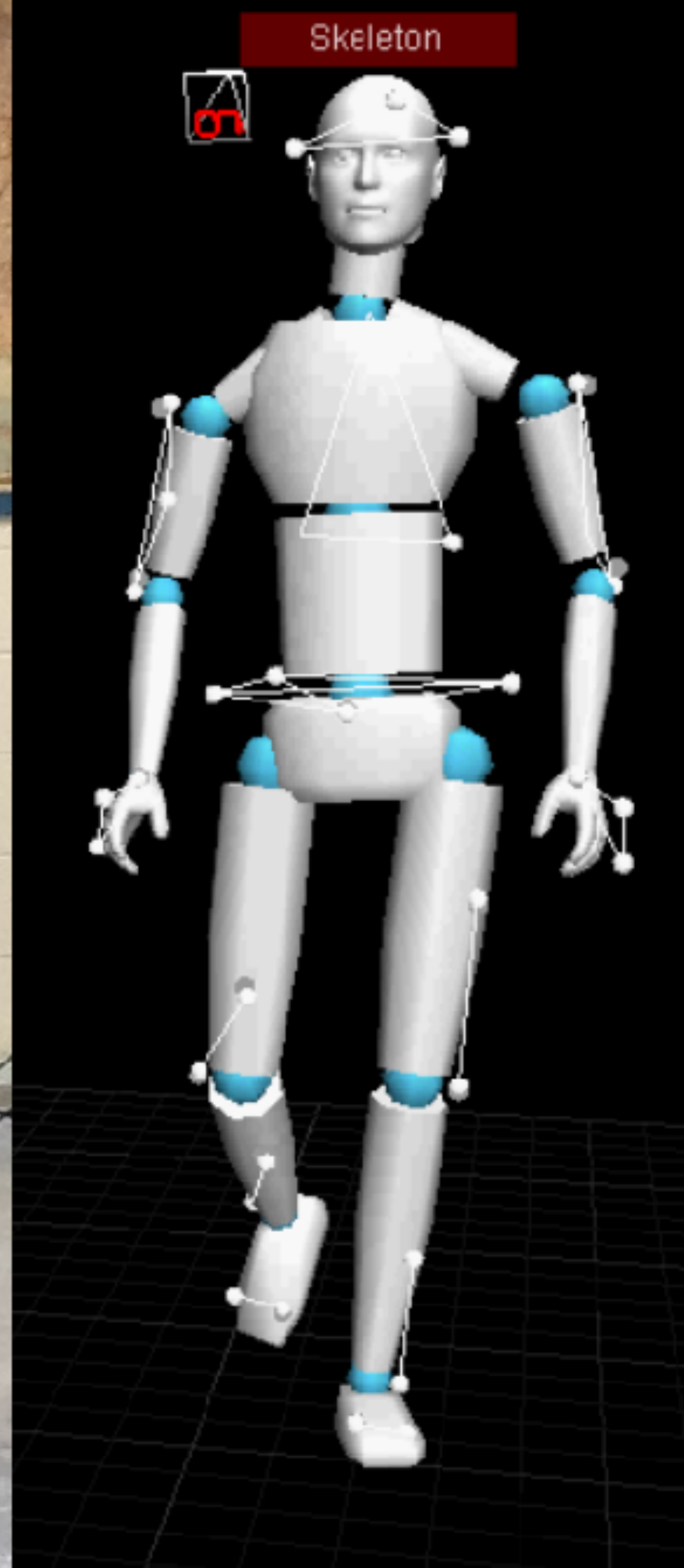

Character Animation by Sebastian Lague

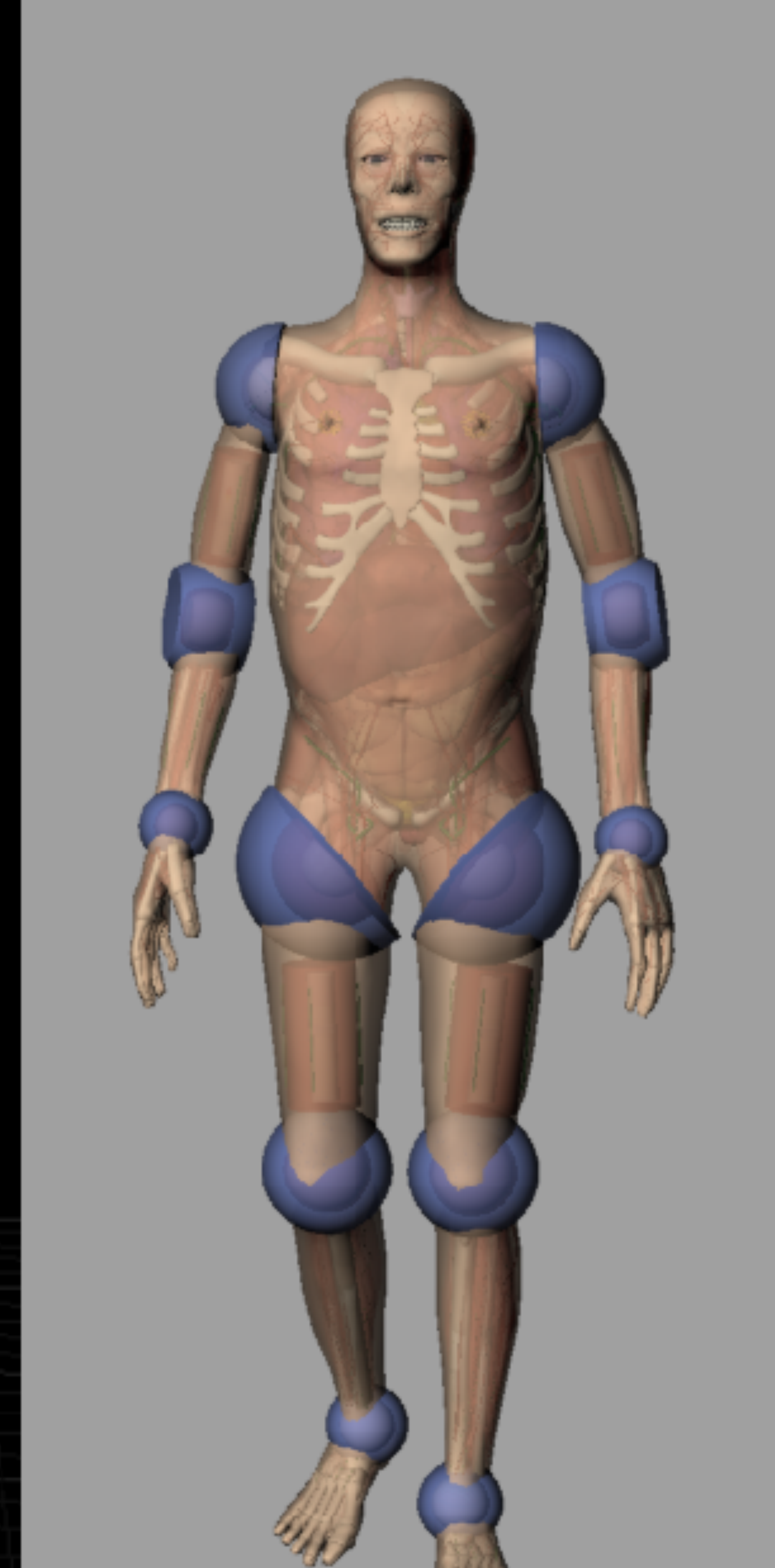

Ed | by Chris Jones

# Motion Capture

# Skeleton Animation

- Human (or animal) movements can be subtle, yet incredibly expressive
- Animating natural movement requires time, effort, and much skill
- Motion capture can provide a more efficient solution to this problem



[image by Vazquez88]


Motion Capture: Avatar

Twentieth Century Fox

# Things to Remember

- The goal of computer animation is to impart life, interest, spirit, motion, or activity into your scene
- Many techniques exist for accomplishing this goal:
    - keyframe, procedural, physics-based
- Characters are particularly challenging to animate
    - motion capture techniques can help immensely
- Apply the 12 classic principles when appropriate!