Introduction to 3D Computer Graphics

Pisut Wisessing
pisut.wi@ku.ac.th

2023-03-21

Slides were originally created by Kerstin Ruthland and John Dingliana, Trinity College Dublin, Ireland

Pisut Wisessing

พิศุทธิ์ วิเศษสิงห์

- Education:

  PhD in Computer Science

  Trinity College Dublin, Ireland

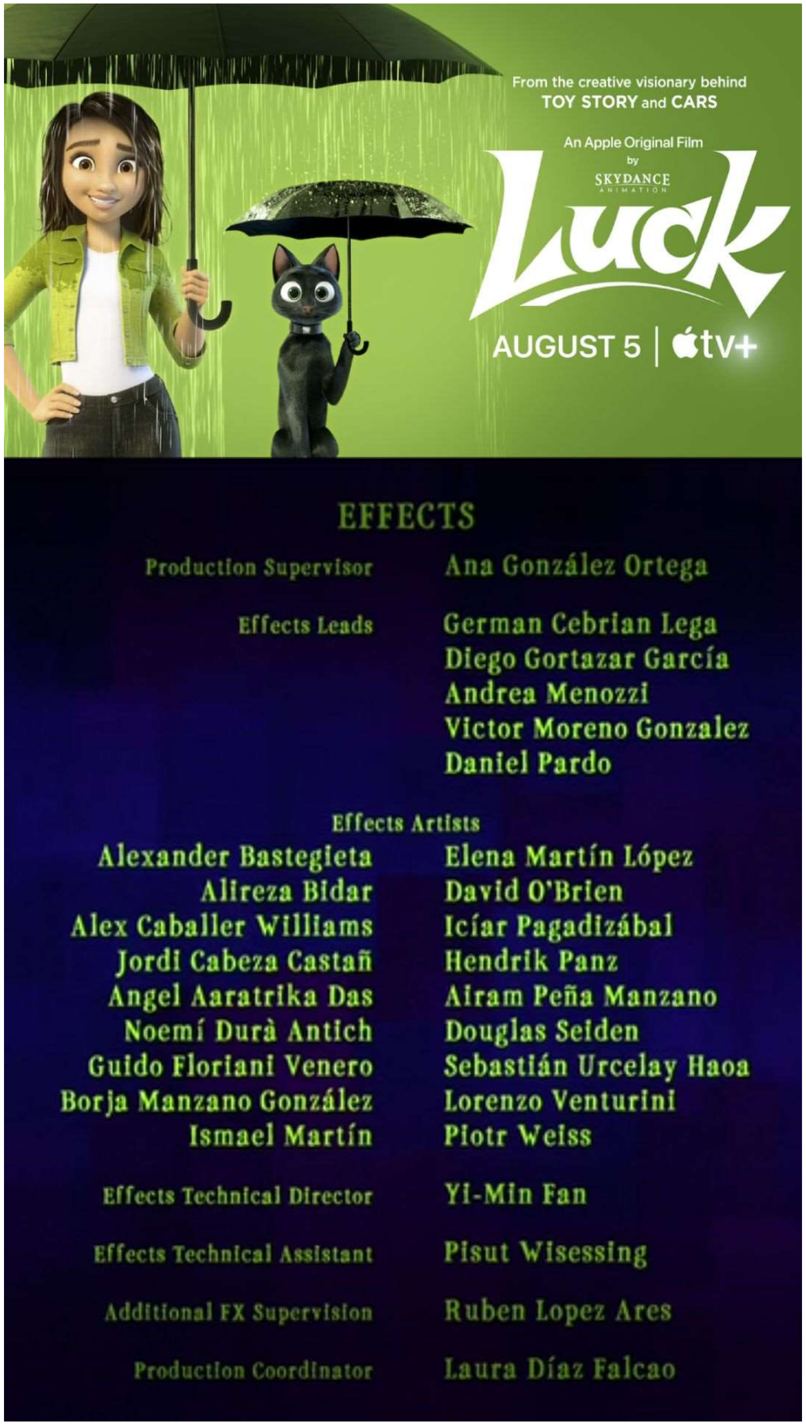
  MFA in Digital Production

  Clemson University, USA
- Current Position:

  Lecturer in Computer Science
- Office:

  Room: SC45-771
- Email:

  pisut.wi@ku.ac.th



Recommended Online Resources

Khan Academy's Pixar in a Box

www.khanacademy.org/computing/pixar

- A good overview of CG animation production process (artistic and technical)

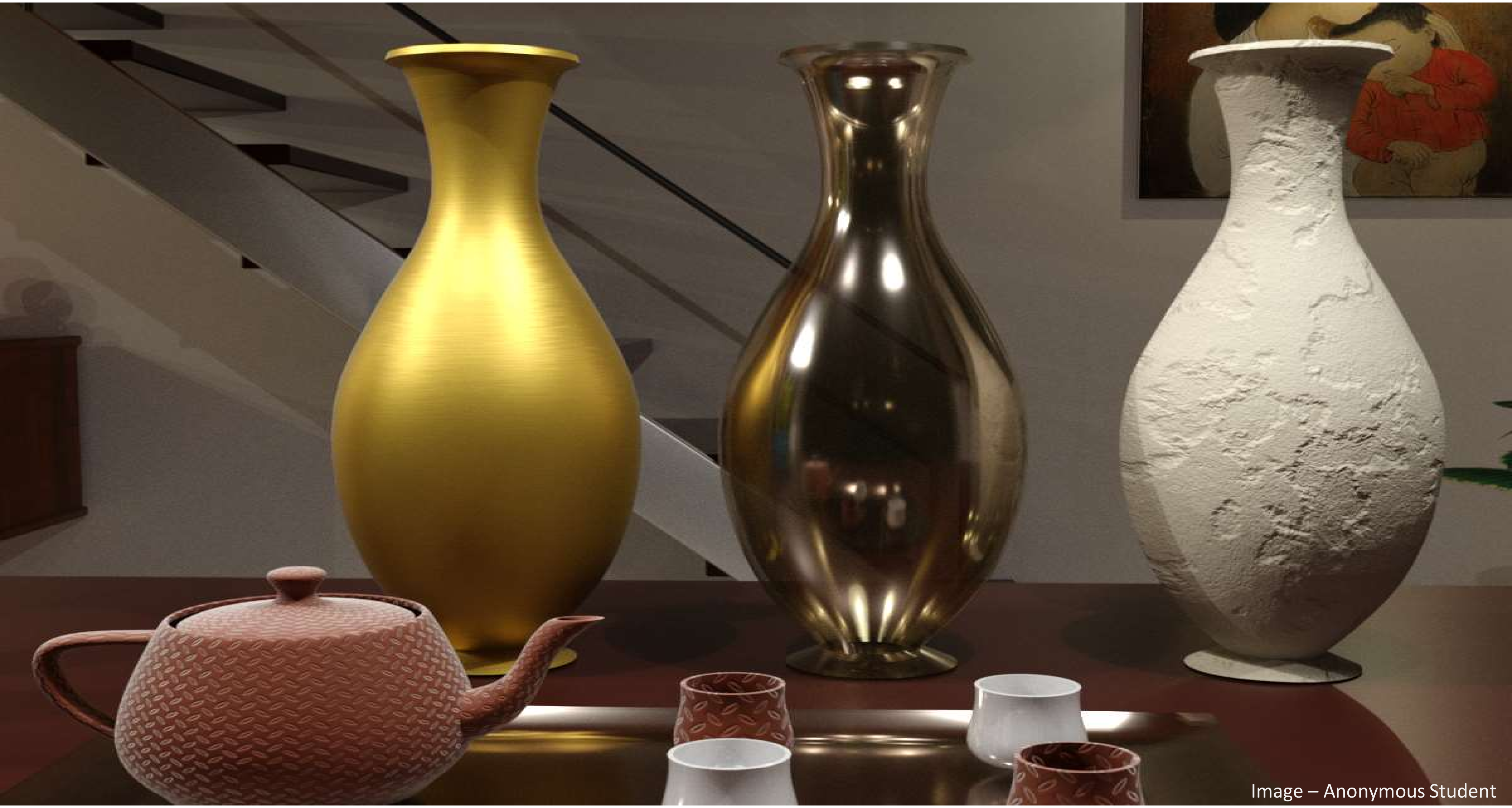
Blender Online Tutorials

www.blender.org/support/tutorials/

- Blender Fundamentals 2.8x (similar to 3.3x)

Applications of 3D Graphics

Motivation for visual computing



Computer Animation

CG as a medium

Pixar
(1995)

Illumination
(2010)

Walt Disney
(2013)

Netflix / DreamWorks Animation
(2016 – 2018)

DreamWorks Animation
(2014)

Computer Games

Epic Games (2017 -)

Ustwo Games (2014 & 2017)

CD Projekt (2020)

Visual Effects

Guardians of the Galaxy Vol. 2 – Walt Disney Studios (2017)

Virtual Production

The Mandalorian – Lucasfilm / Disney (2019)

Virtual Reality (VR) & Augmented Reality (AR)

The Sword of Damocles (1968)

Beat Saber (Quest 2)

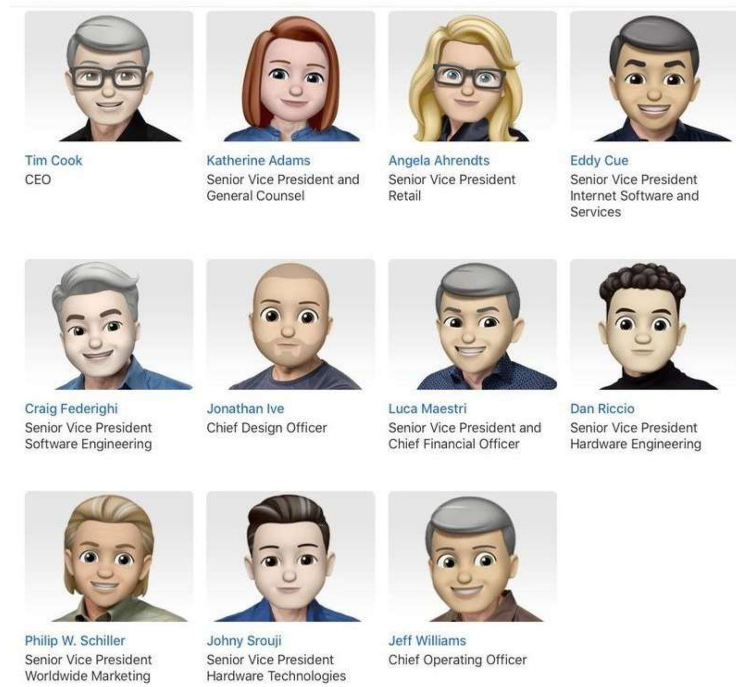
HoloLens 2

Magic Leap

Pokemon Go

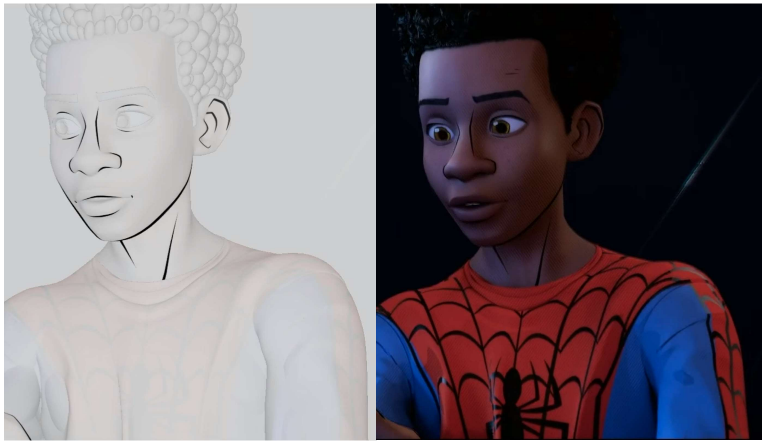
Social Media & Metaverse

Meta Horizon Worlds



Apple Memoji

2D/3D Hybrid Animation



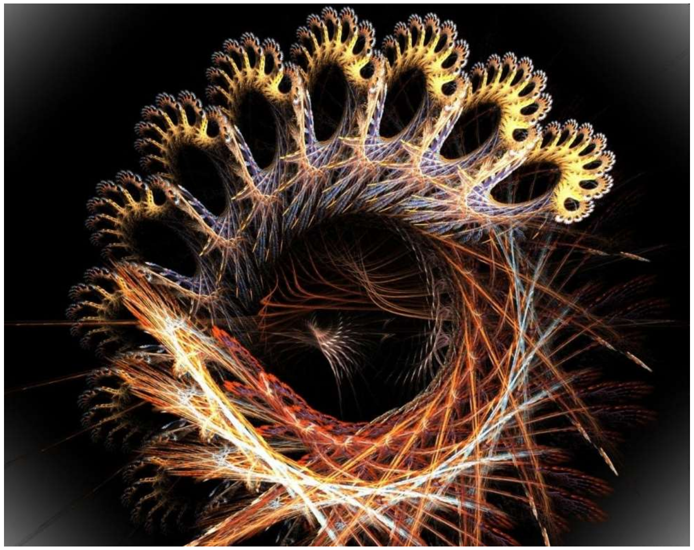
Spider-Man: Into the Spider-Verse – SPI (2018)

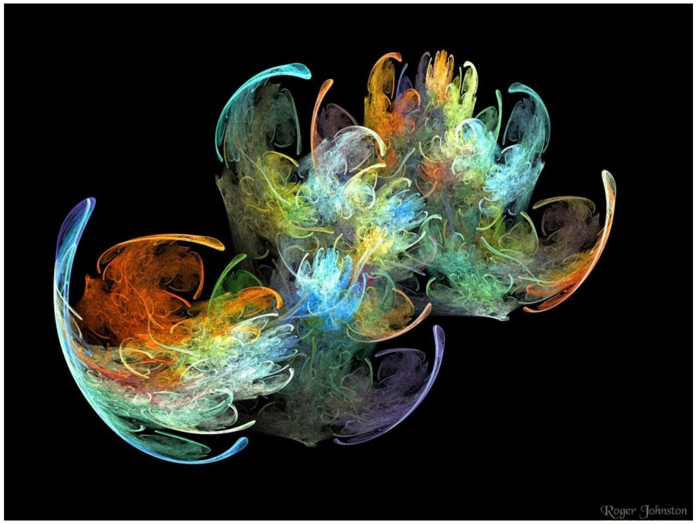
Klaus – Netflix (2019)

- 3D animation
- Comic-book lines and textures

- 2D animation
- Computer-aided shading to mimic 3D lighting

Abstract Art and Animation

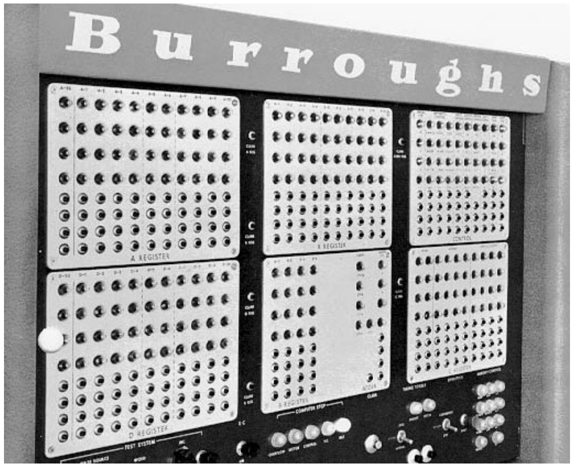




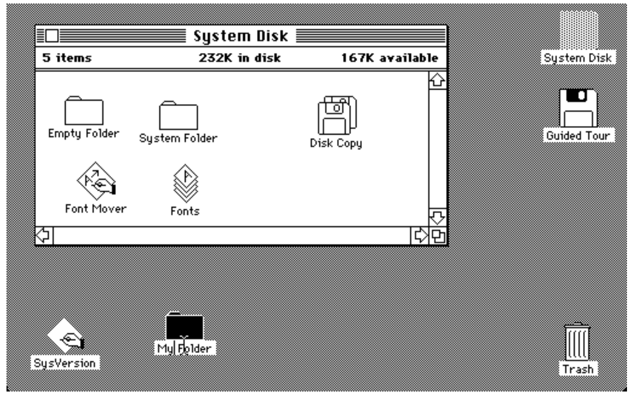
Anjyu - Yasuo Ohba (2001)

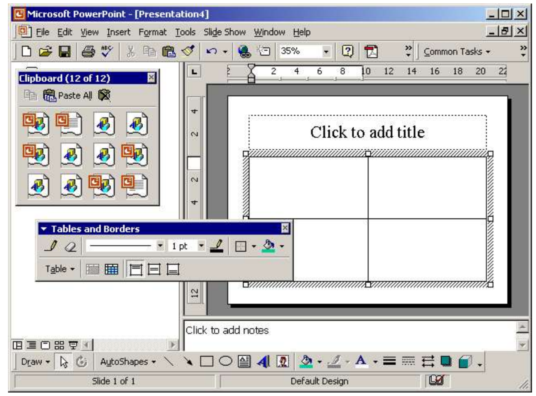
User Interfaces

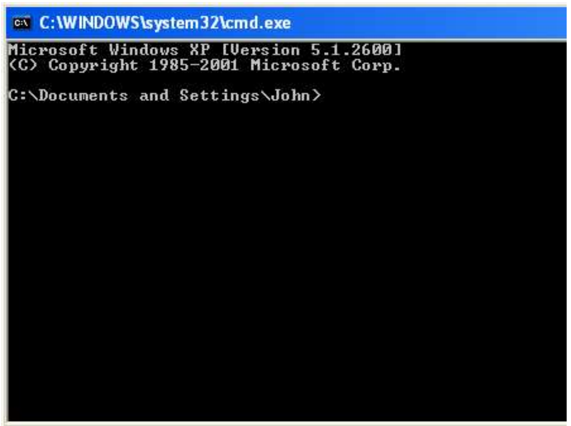
How would we communicate information from Machine to User if not for Imagery



B205 Control Console (1960)



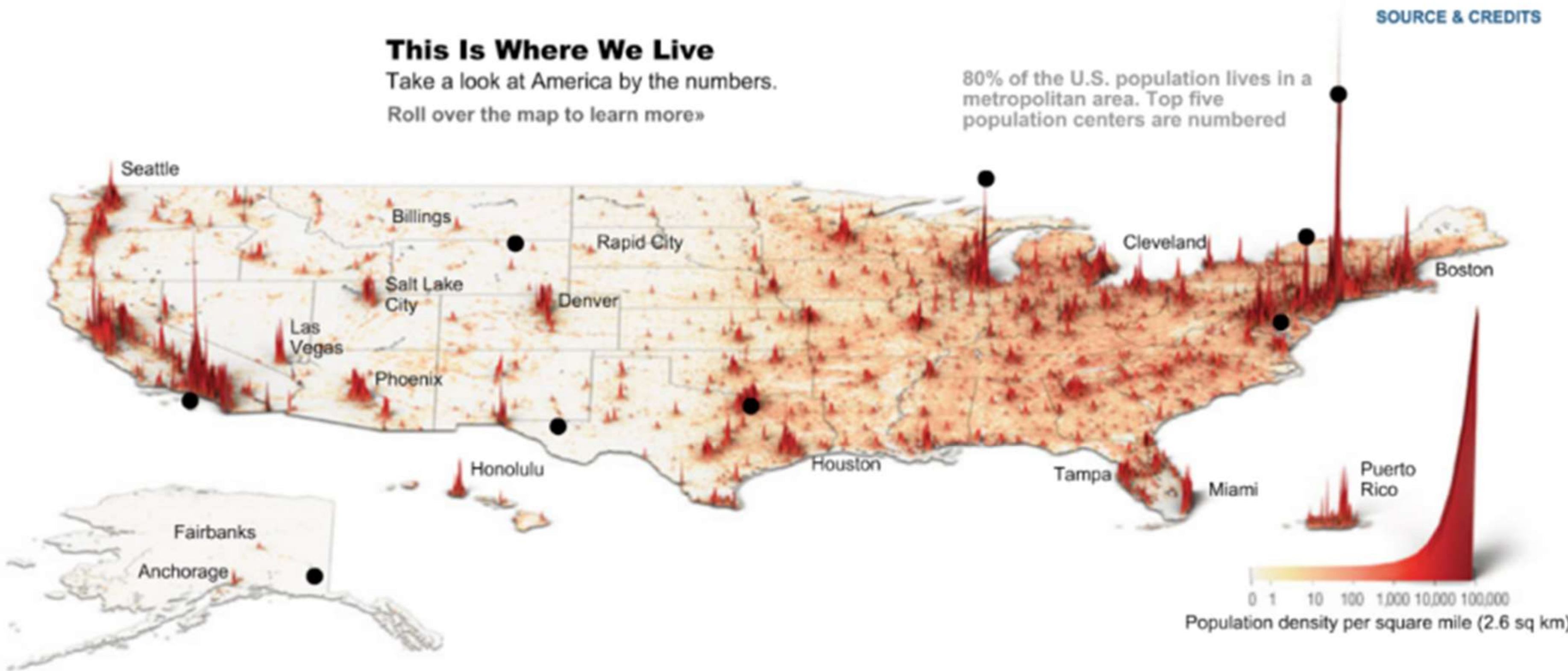




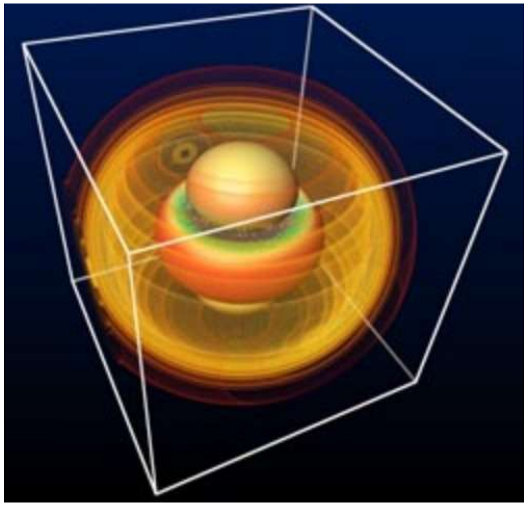
A more modern “console” (2000)

Graphical Interfaces

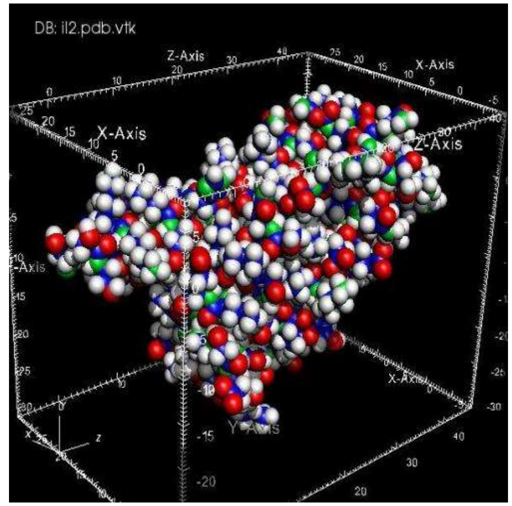
Information Visualization



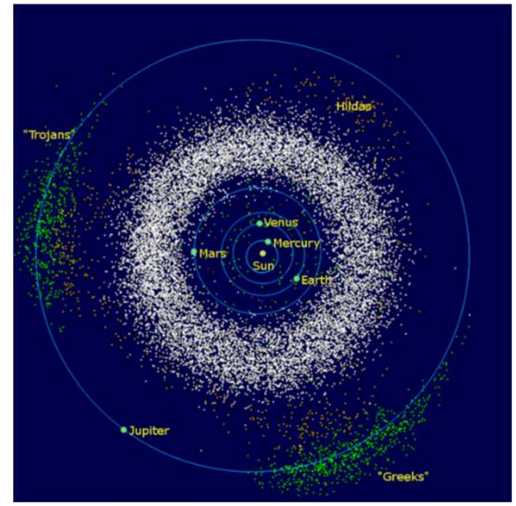
Scientific Visualization



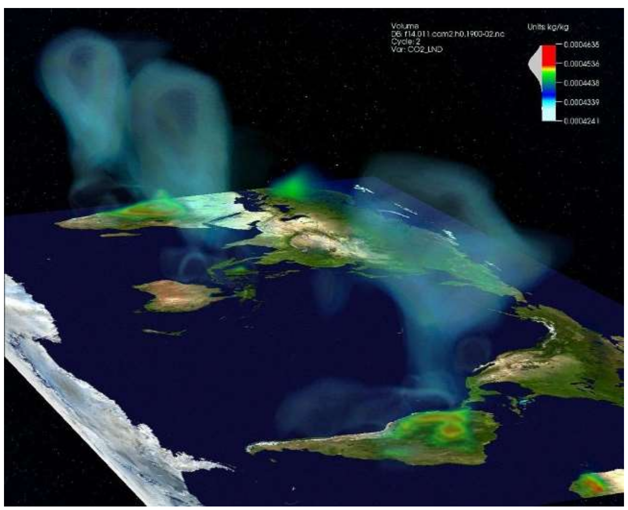
Gravity Waves (I. Foster, C. Kesselman & S. Tuecke)



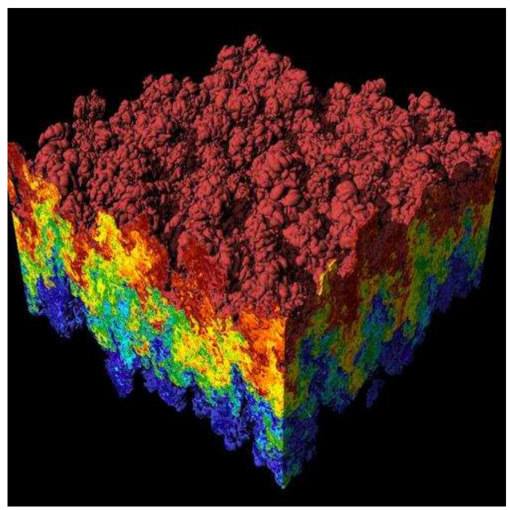
Molecular Visualization (UCRL-WEB)



Solar system, by Mdf.

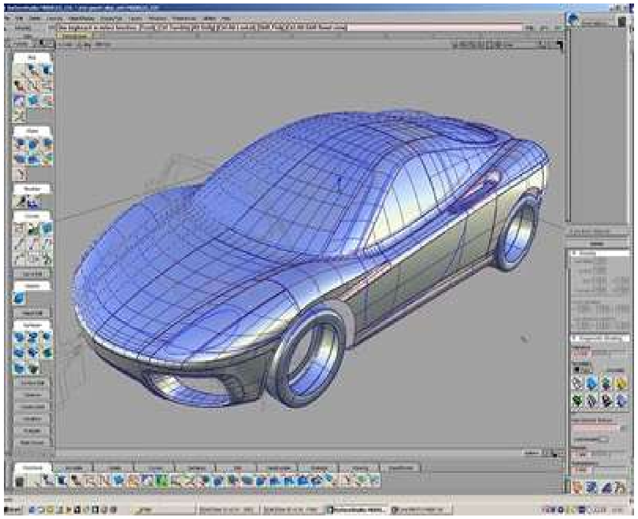


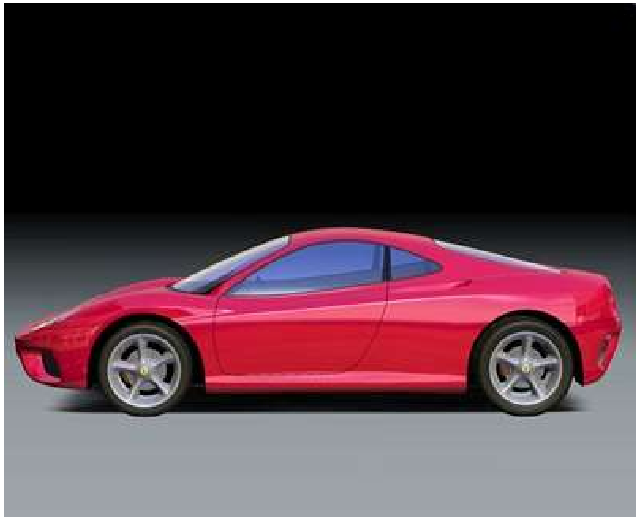
Climate visualization (UCRL, F. Hoffman & J. Daniel of Oak Ridge National Laboratory)



Fluid simulation (Lawrence Livermore National Laboratory)

Commercial Design and Prototyping





Computer Images

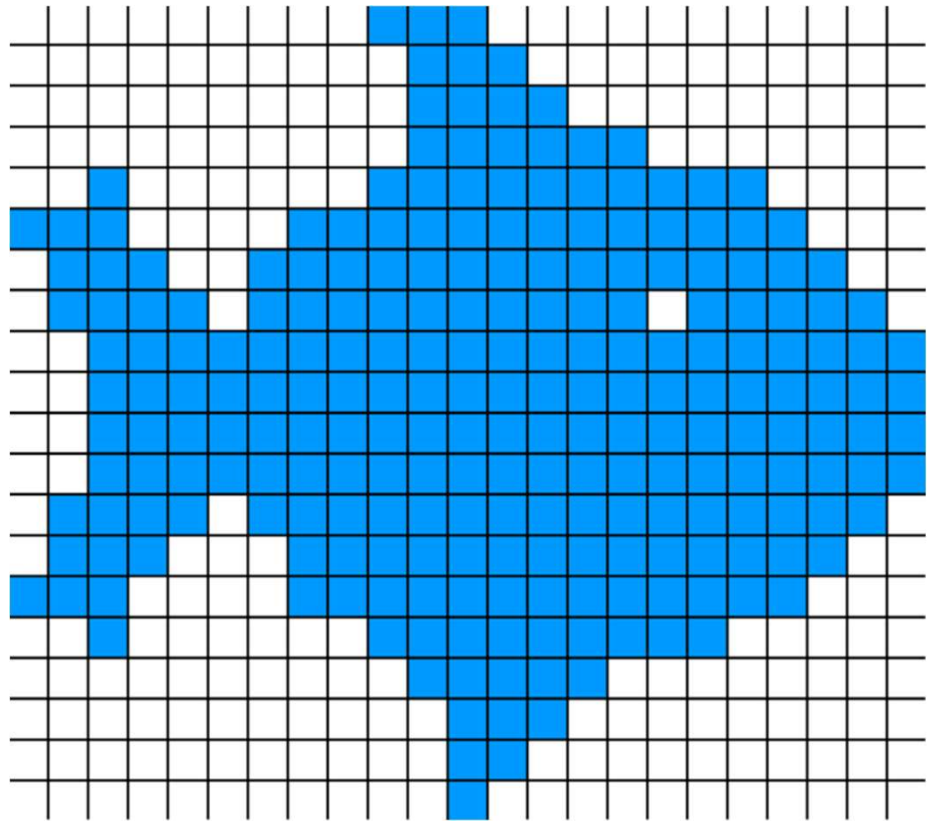

Image – Anonymous Student

Digital Images

– A visual output of data stored in terms of numeric, recordable elements.
– All of these are represented by (whole) numbers: i.e. digital.

Raster (Pixel) Images

Most digital images we use today are output as a regular grid of pixels, referred to as a **raster.**



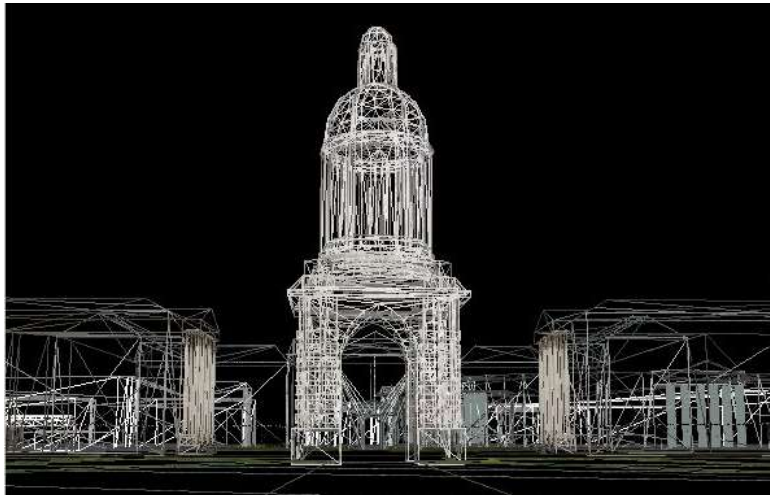
Andreas Horning © Used under creative commons

Modelling

Image – Anonymous Student

Modelling

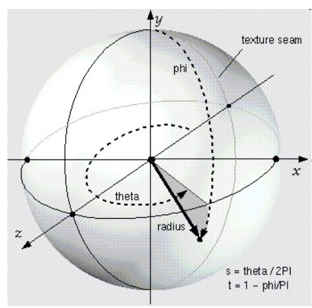
Model: an abstract or actual representation of a system or object. Models in Computer Graphics deal with data that eventually leads to graphical output such as still and moving images.

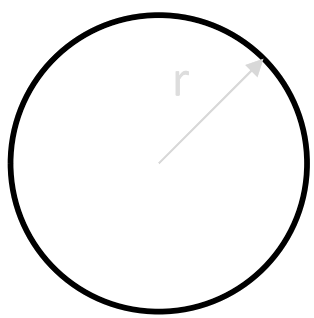


Levels of Modelling

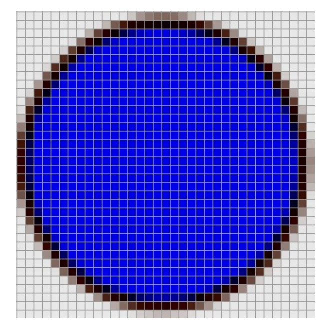
The more abstract the representation, (arguably) the better our opportunities for re-usability and data/computational efficiency.

3D / 2D Object

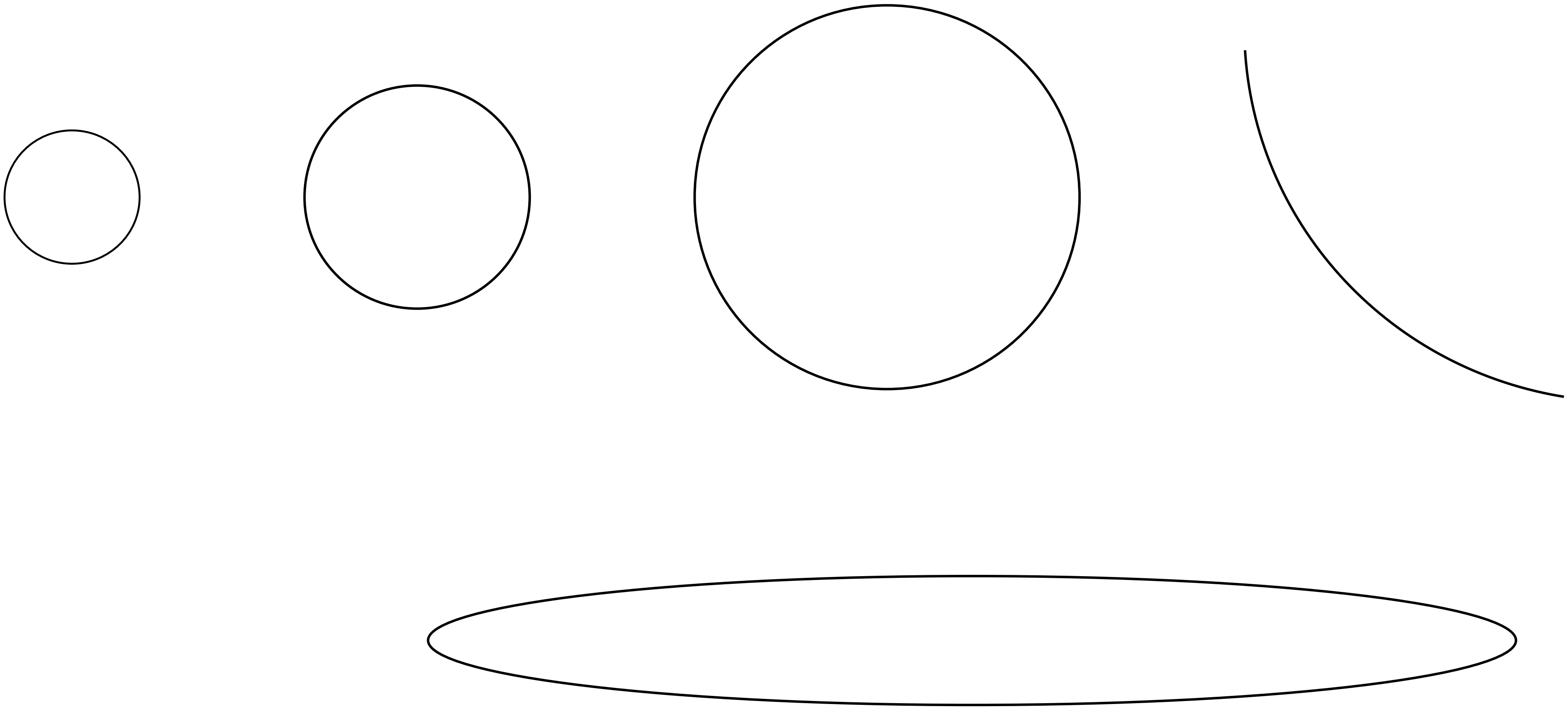




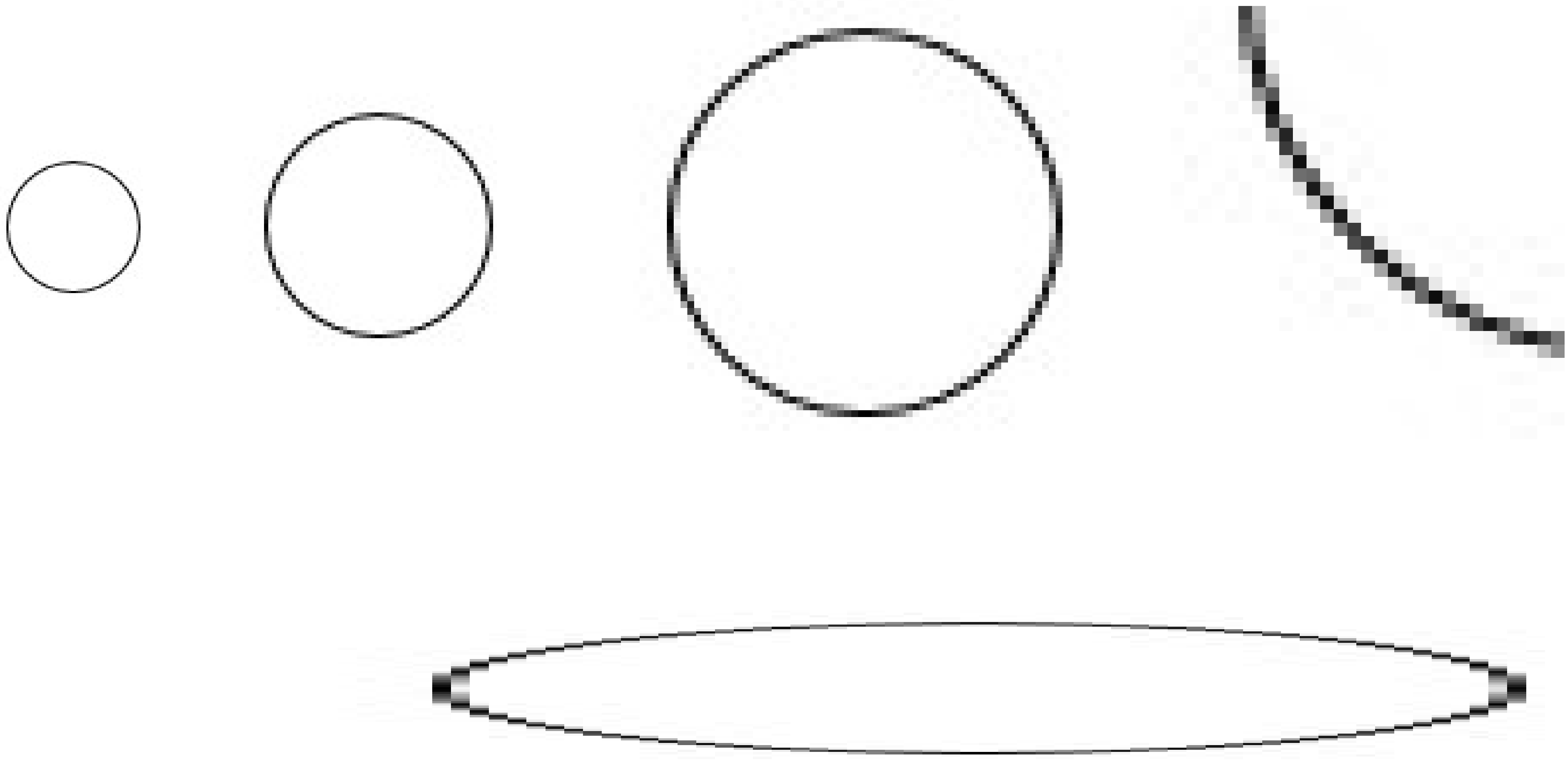
Image



Levels of Modelling – 2D Object



Levels of Modelling – Raster (image)



Models vs. Images

processed / rasterized /rendered

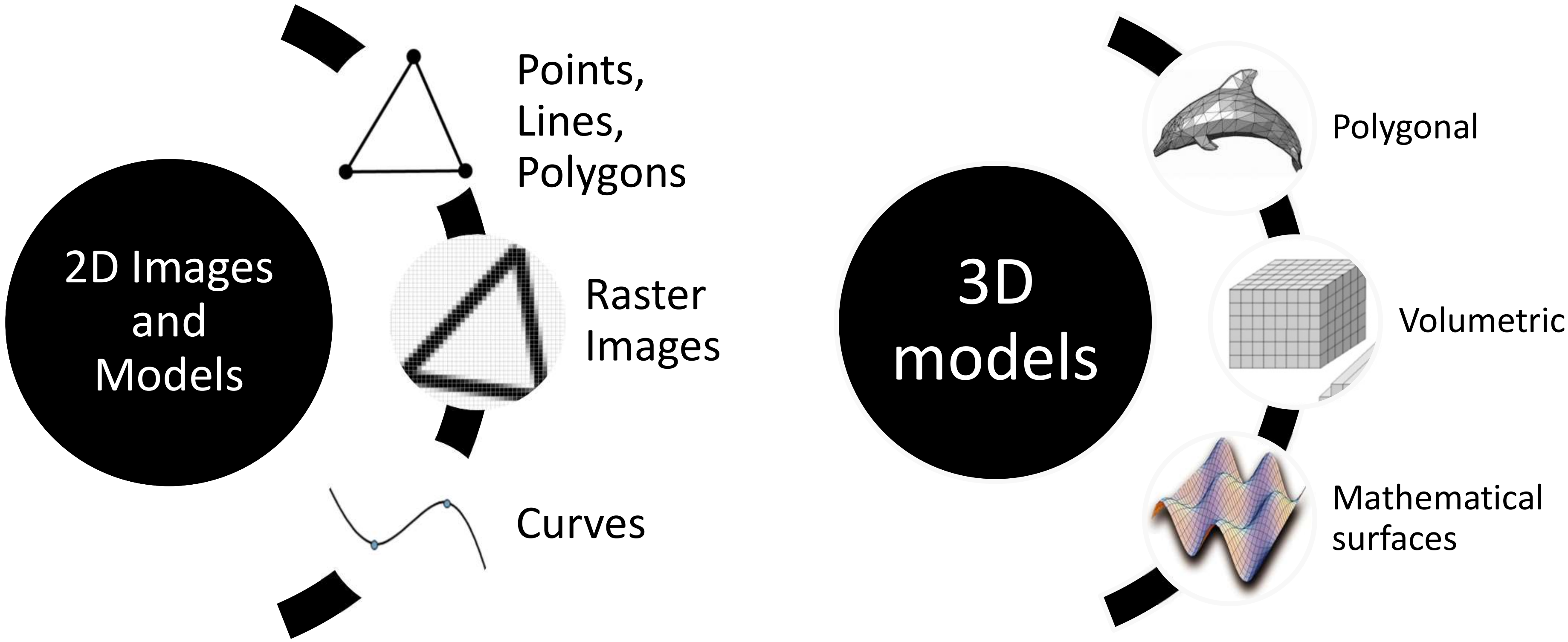
Models → **Images**

Models	**Images**
– More abstract	– Low level
– More compact	– Resolution bound
– More Reusable	– Less re-usable
– Easily Modifiable	– More limited to modification

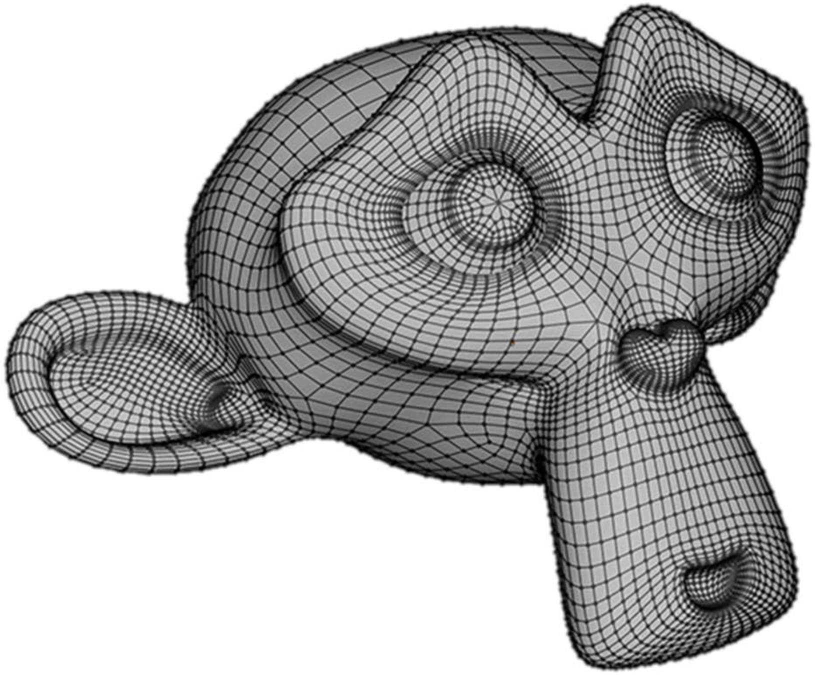
Modelling 3D Objects

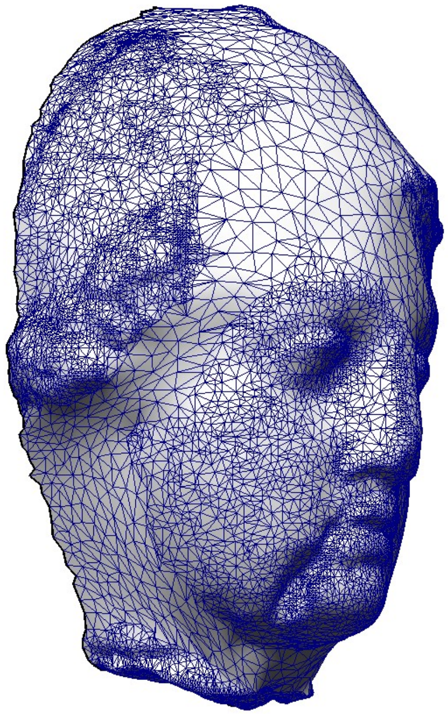
Image – Anonymous Student

Modelling Techniques



Polygon Mesh





This is the basis for modelling a large range of complex shapes in 3D applications

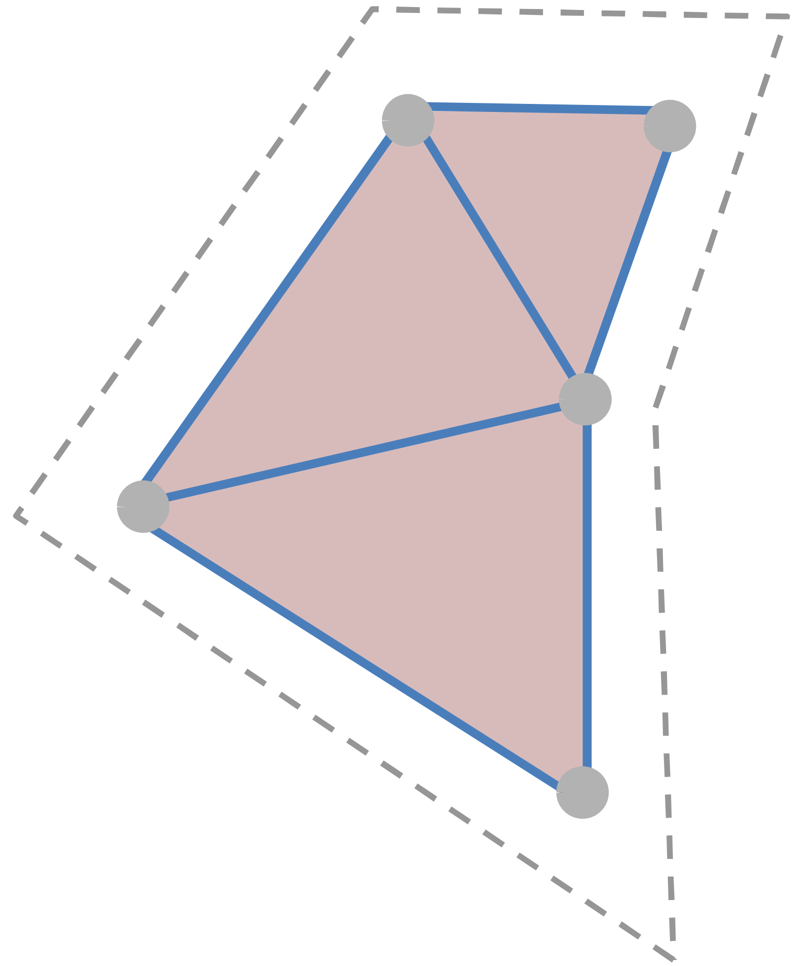
Representing objects with Polygons

1) Start with Points/Vertices representing the surface of an object.

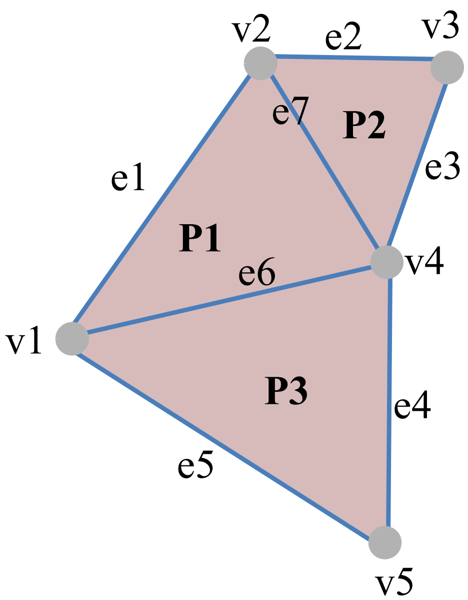
– Defined as positional vectors *<x,y,z>* from the origin

2) Connect points to form Edges (line segments) enclosing Polygons.

3) Connected polygons are called Meshes.



Polygon Mesh



Polygon List

P1 →	e1, e6, e7
P2 →	e7, e3, e2
P3 →	e5, e4, e6

e1 →	v1, v2
e2 →	v2, v3
e3 →	v3, v4
e4 →	v4, v5
e5 →	v5, v1
e6 →	v1, v4
e7 →	v2, v4

Edge List

v1 →	{x1,y1,z1}
v2 →	{x2,y2,z2}
v3 →	{x3,y3,z3}
v4 →	{x4,y4,z4}
v5 →	{x5,y5,z5}

Vertex List

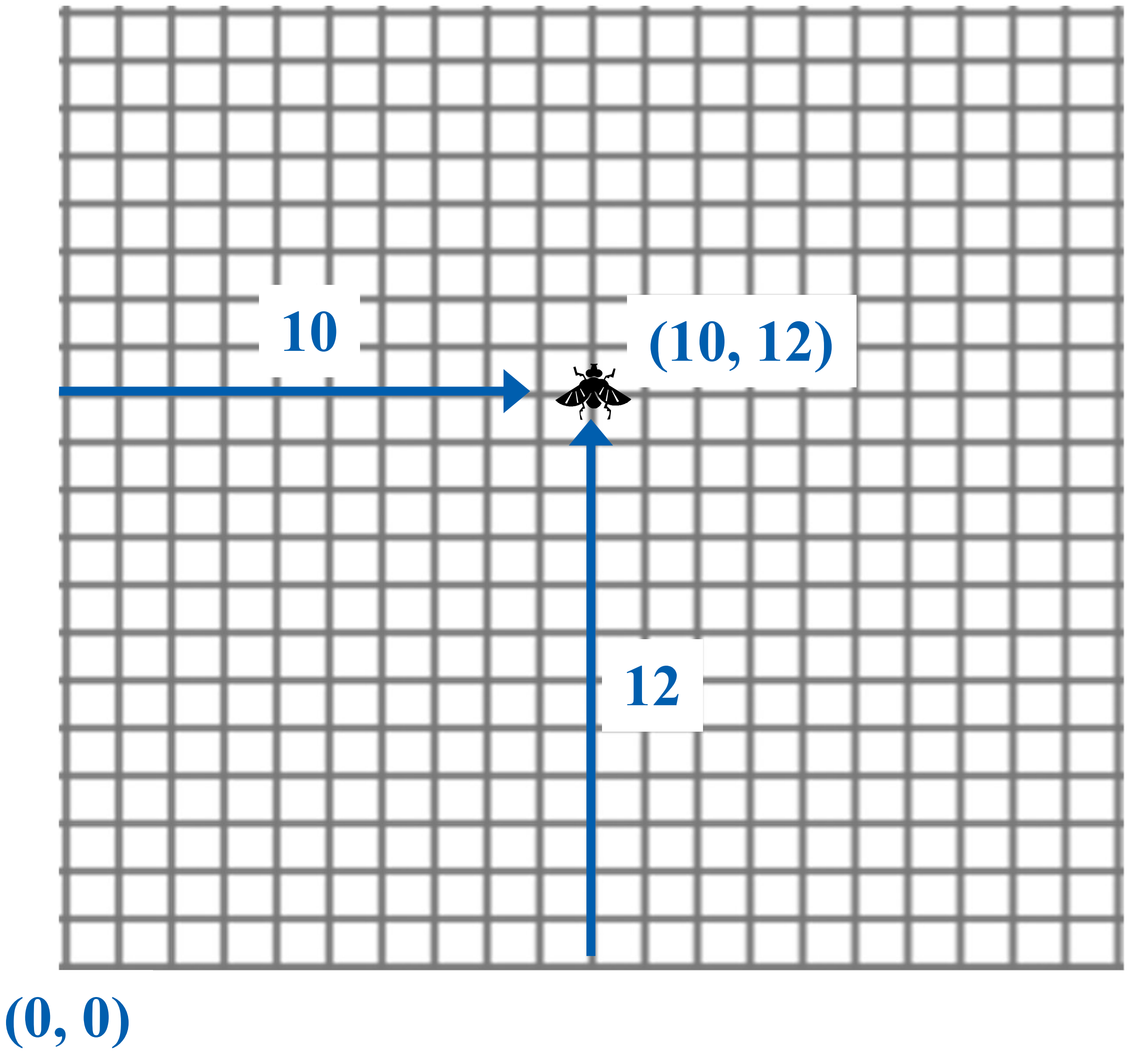
The building blocks of a model

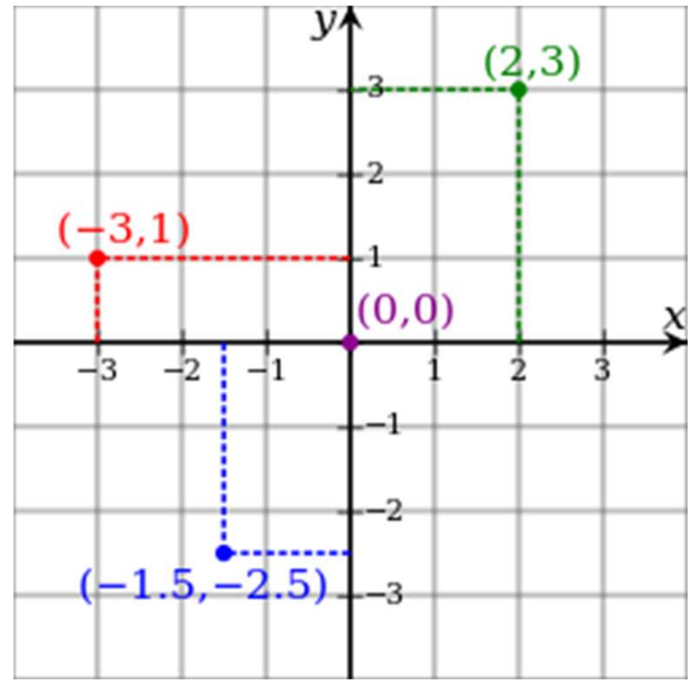
Geometry in computer graphics is most often represented in terms of very basic numerical data

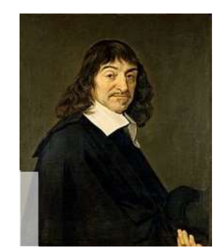
- *Scalars*: single values
- *Vectors*: a couple or triple (or n-tuple) containing scalars – the number of scalars is the dimensionality.
  - Vectors usually represent direction and magnitude

In Graphics, more complex geometry e.g. Curves and Surfaces can also be defined based on scalars and vectors

Cartesian Coordinates

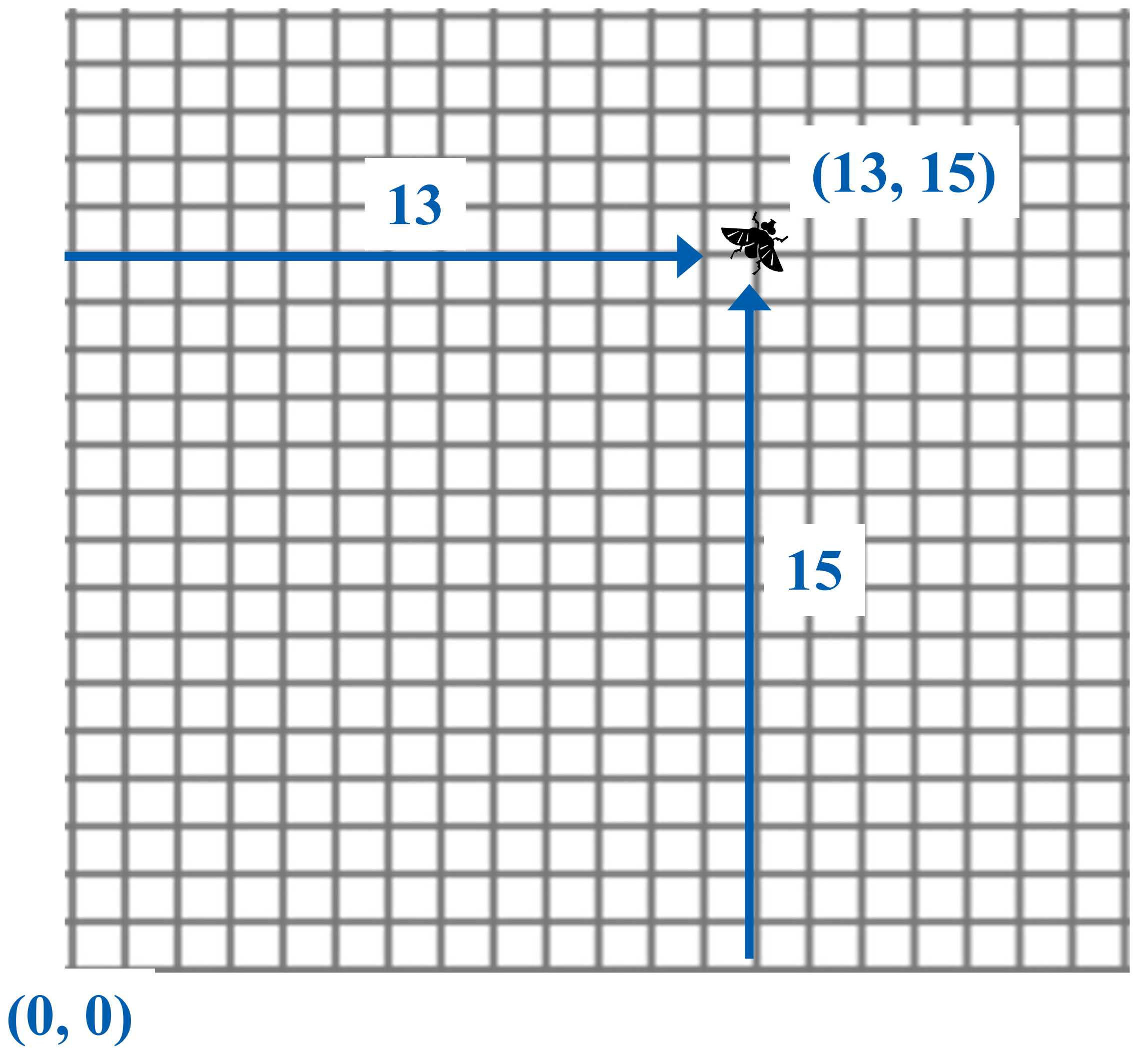






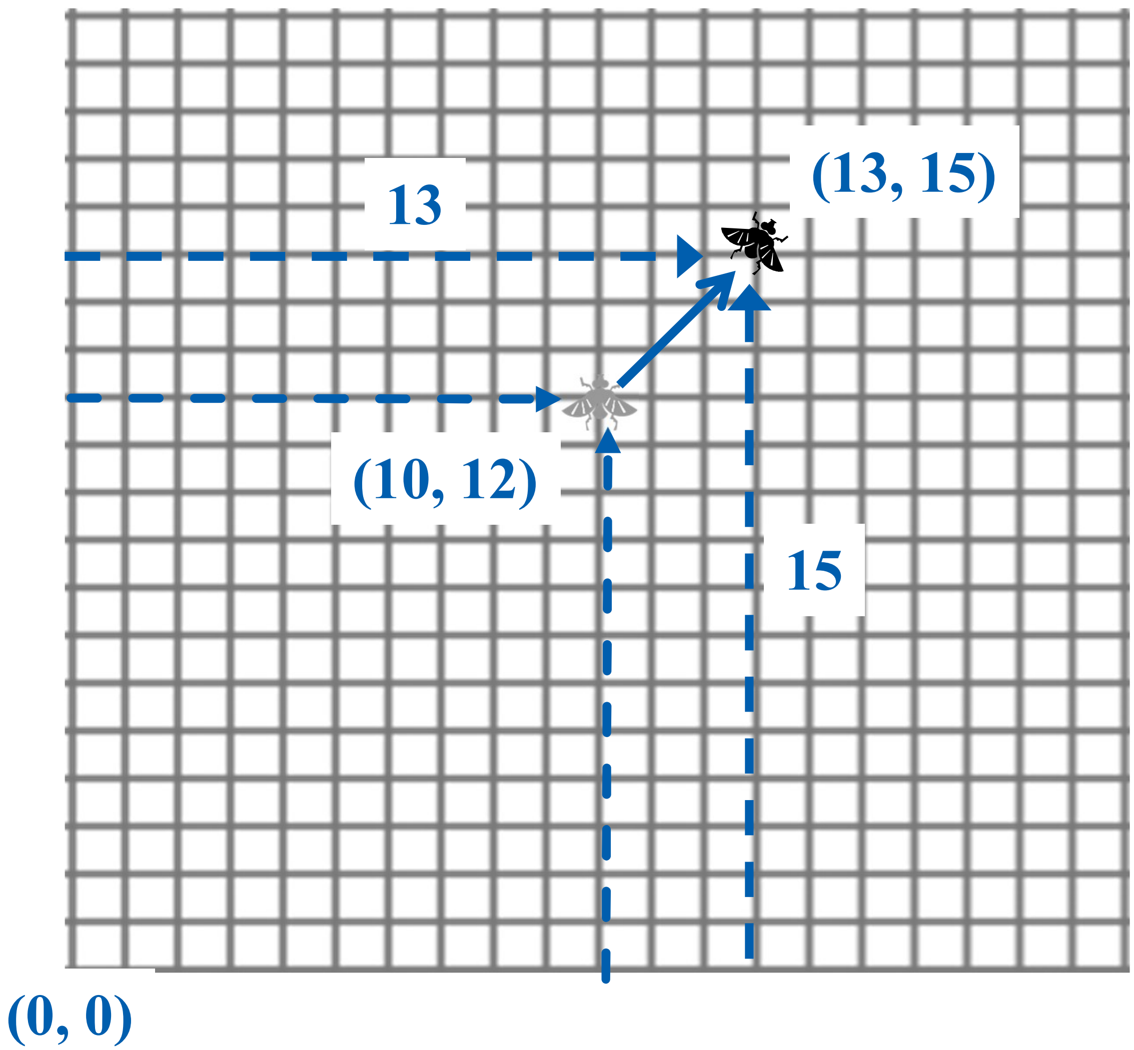
Rene Descartes - 1637

Cartesian Coordinates



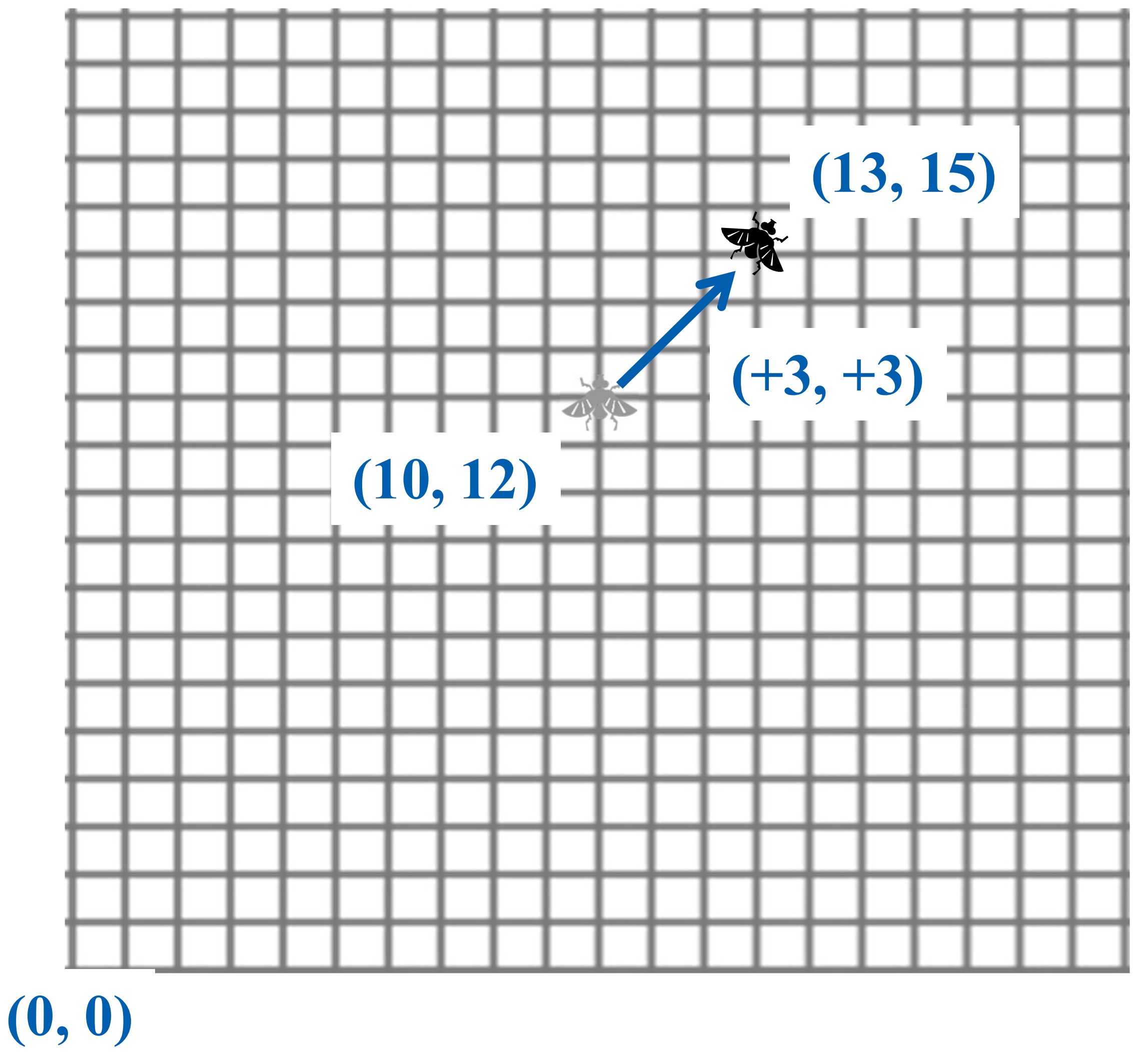
Two distances used to denote unique **position** in 2D

Cartesian Coordinates



We can also represent a line with two **endpoints**, each donated by vector.

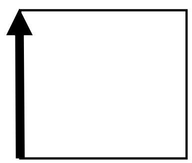
Cartesian Coordinates



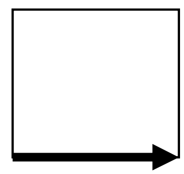
Finally we can denote ***translation*** (or change in position) with two values.

Examples in 2D

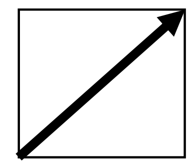
x, y axes



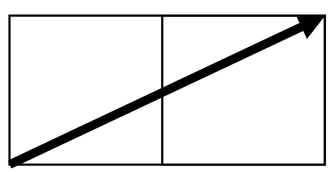
“up” = <0, 1>



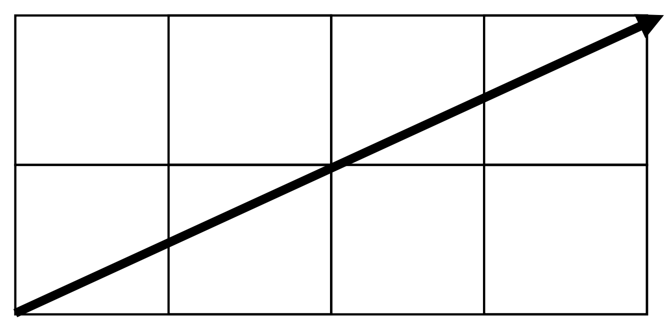
“right” = <1, 0>



North east <1, 1>



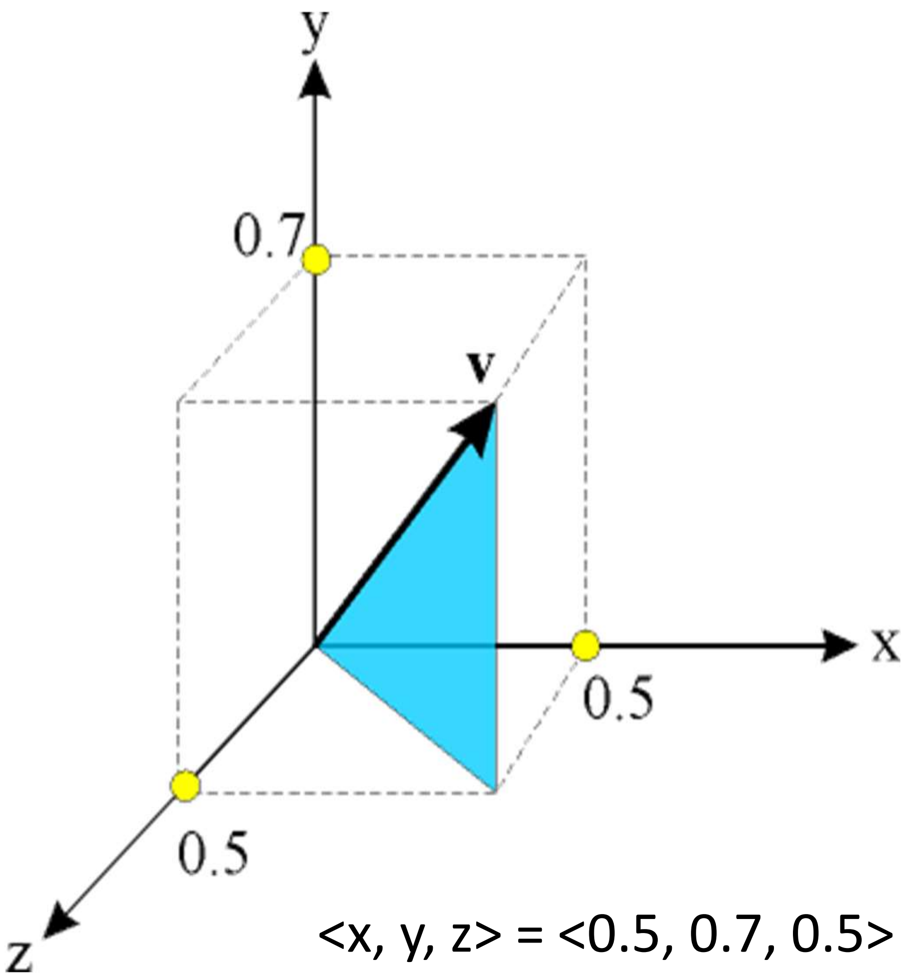
<2, 1>



<4, 2>
Same direction different Magnitude

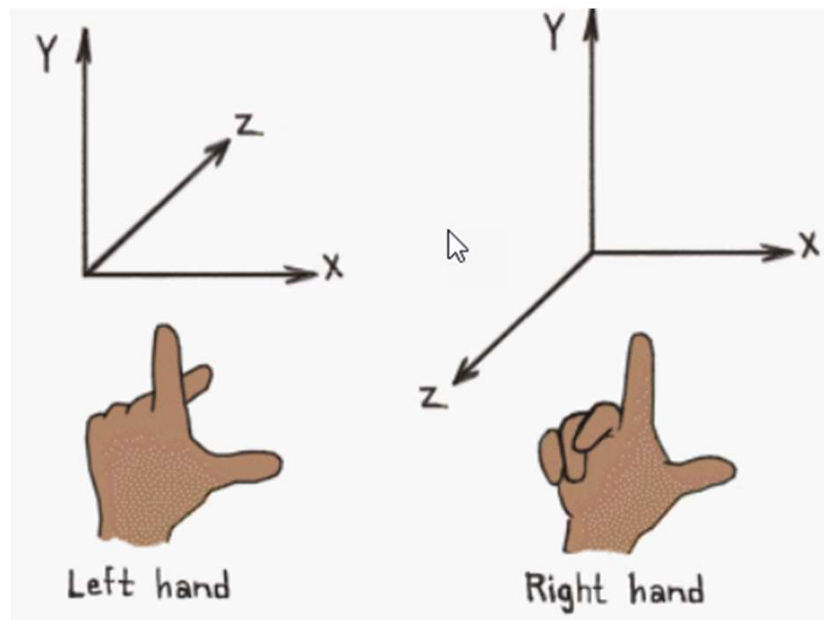
3D Vectors

x, y, z axes

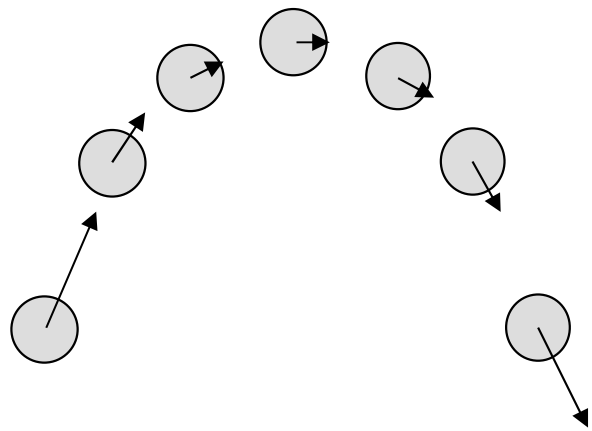


$\langle x, y, z \rangle = \langle 0.5, 0.7, 0.5 \rangle$

N.B. Applications sometimes differ on whether z-axis goes in our out of the screen.



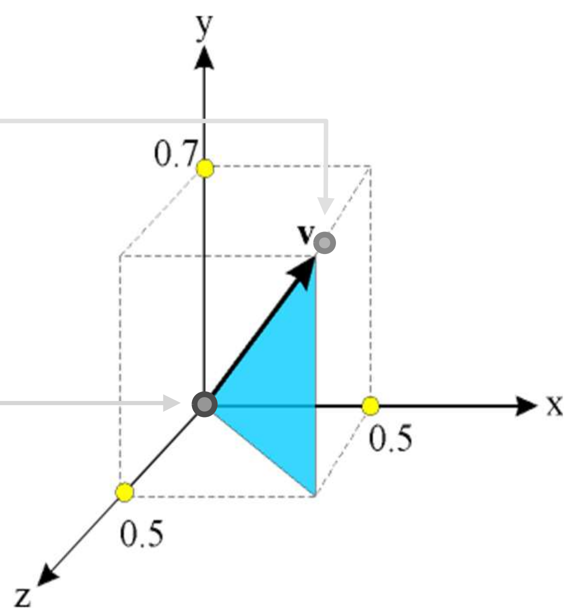
Vectors



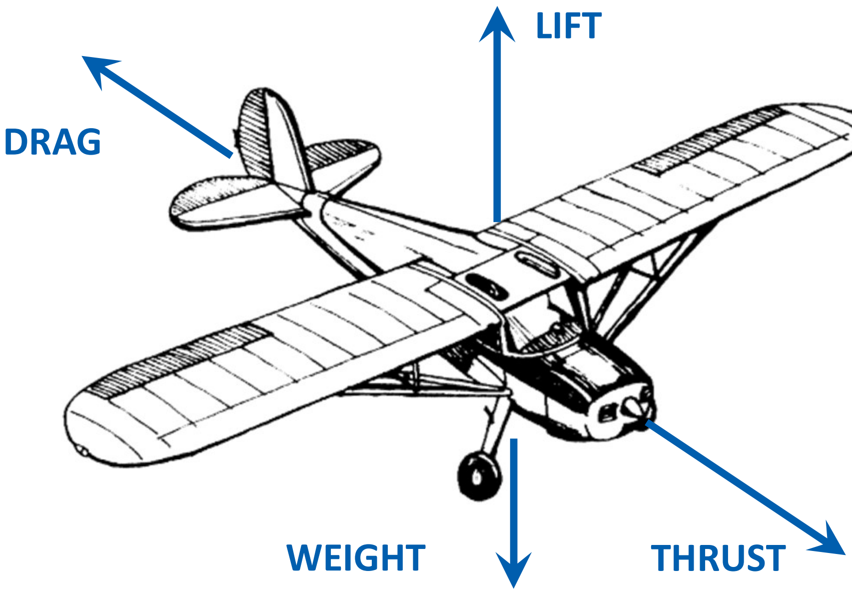
Although used for modelling points, vectors are a lot more general.

Vector **represents**
- a direction
- and a magnitude

3D Point/Vertex
- a location in a 3D space relative to a reference point
- a displacement (vector) from the origin < 0, 0, 0 >
- *Point = Vector + Origin*

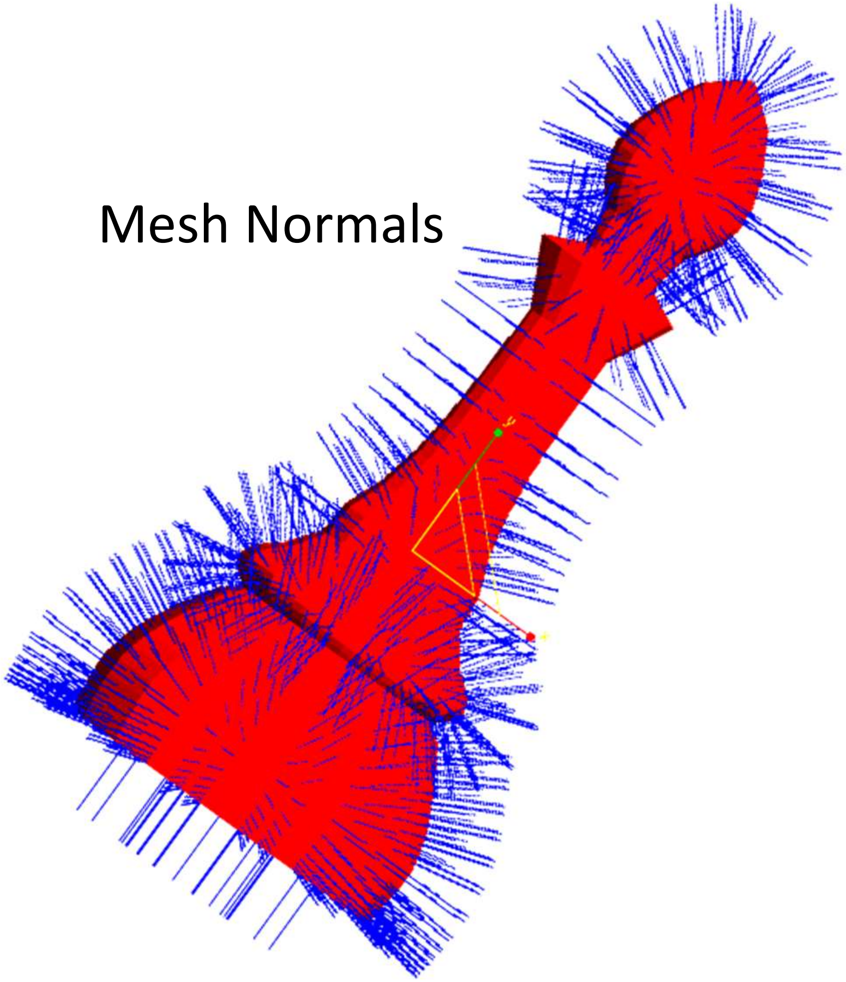


Uses of Vectors in 3D Graphics



The **forces** of flight.

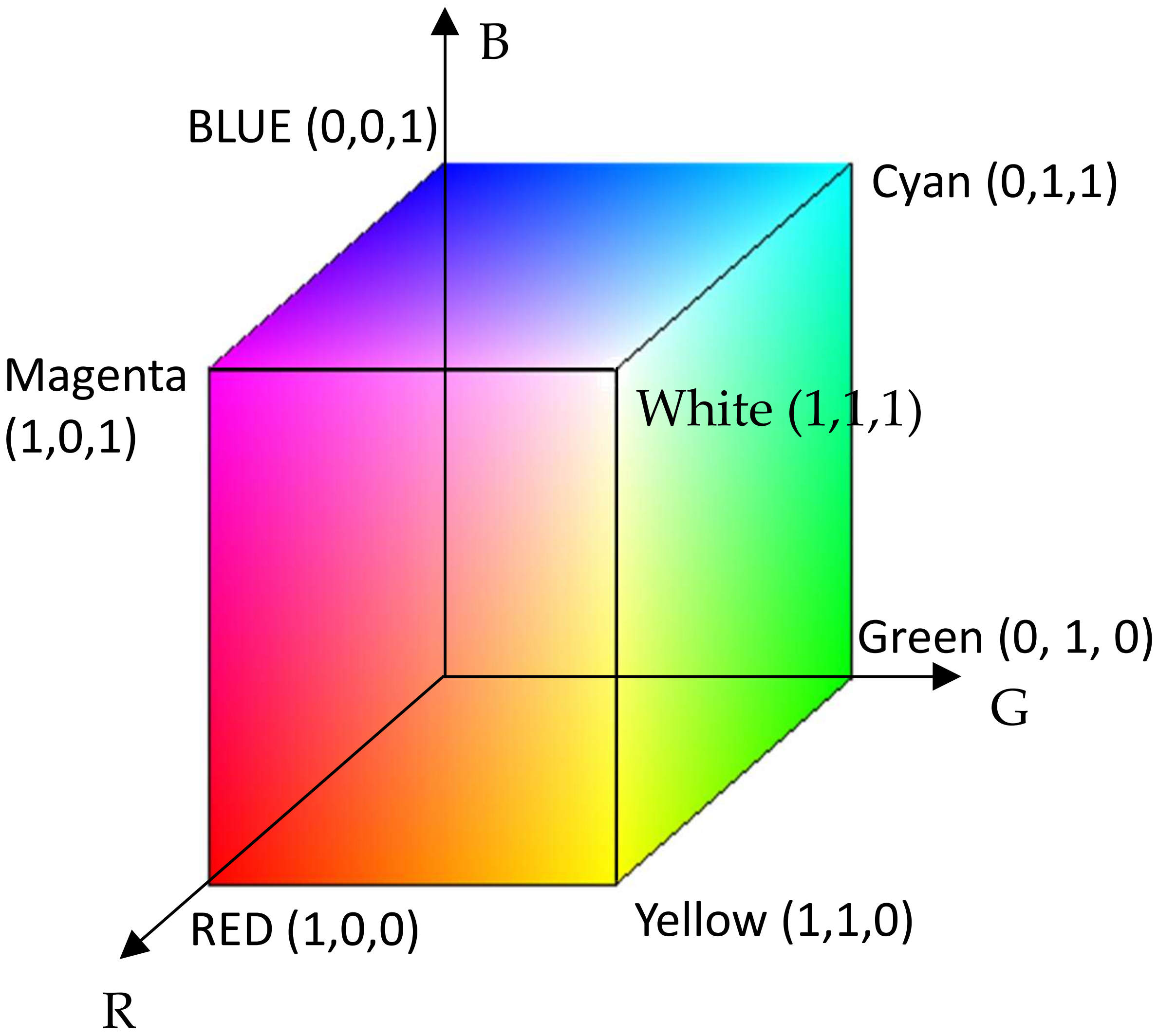
Mesh Normals



Vectors for colour

<red, green, blue>

N.B. Different systems have different ranges for the values:
e.g.
0 to 1:
 full red is <1, 0, 0>,
 mid grey is <0.5, 0.5, 0.5>
0 to 255:
 full red is <255, 0, 0>,
 mid grey is <127, 127, 127>
00 to FF :
 full red is #FF0000
 mid grey is #808080



Recap

Vectors are used for:

- Position <x, y, z>
- Direction
- Axes
- Colour <red, green, blue>

Also used for other things e.g. velocity, acceleration.

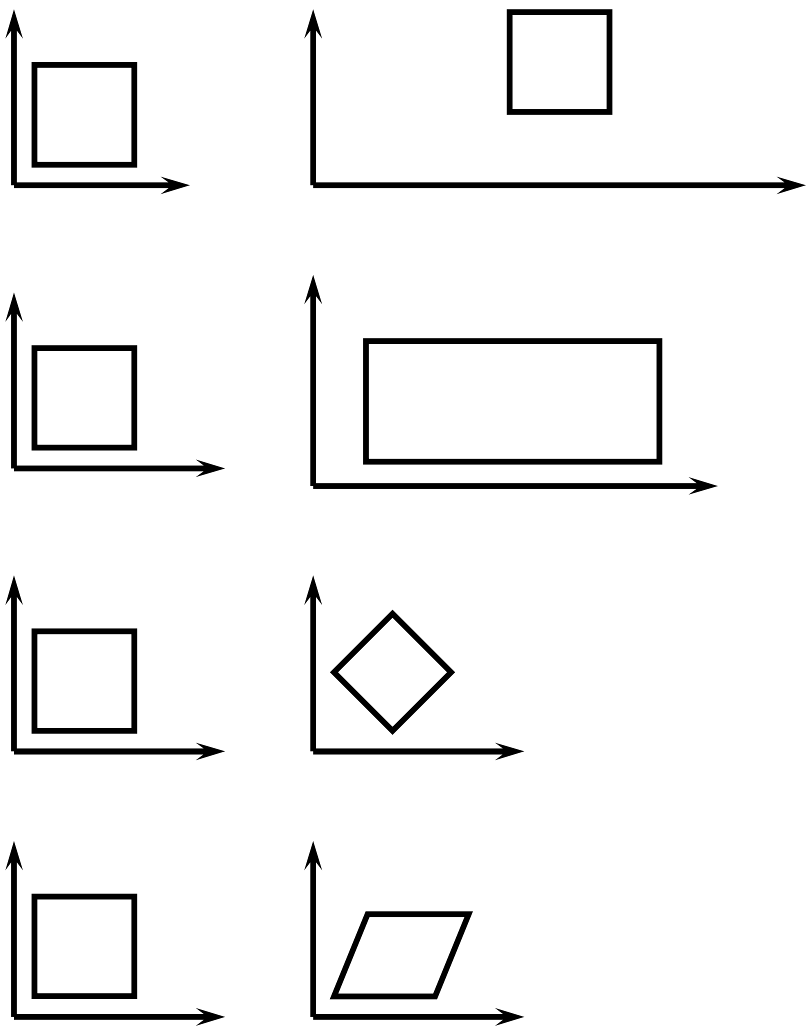
Transforms

Rigid Transformations

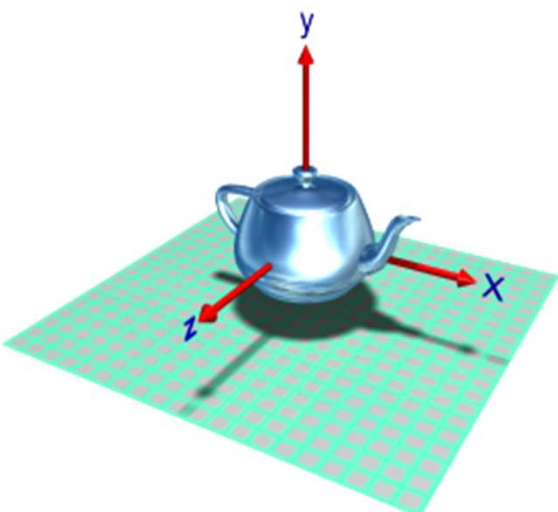
- Translate
- Scale
- Rotate

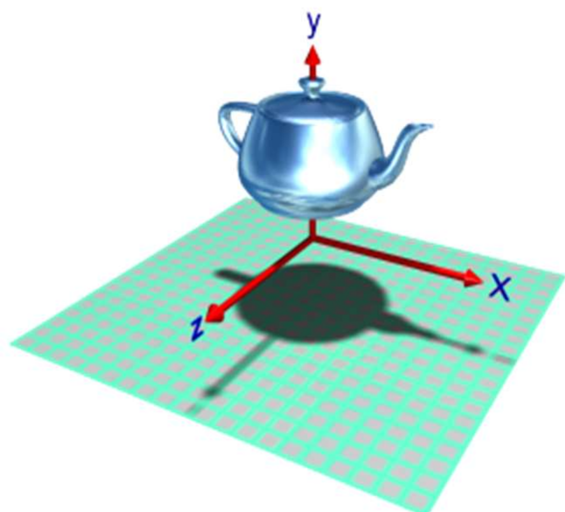
Other transformations

- Shear



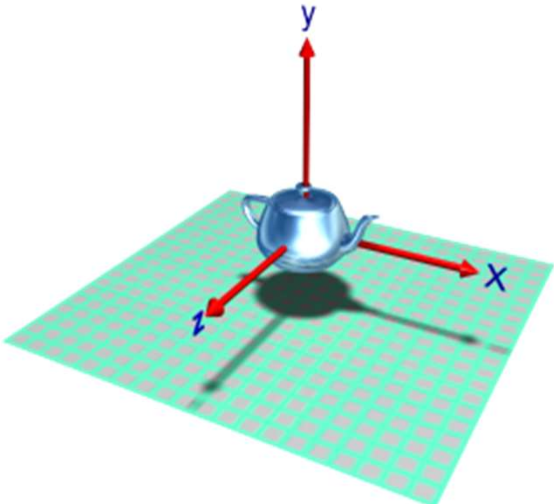
Translation



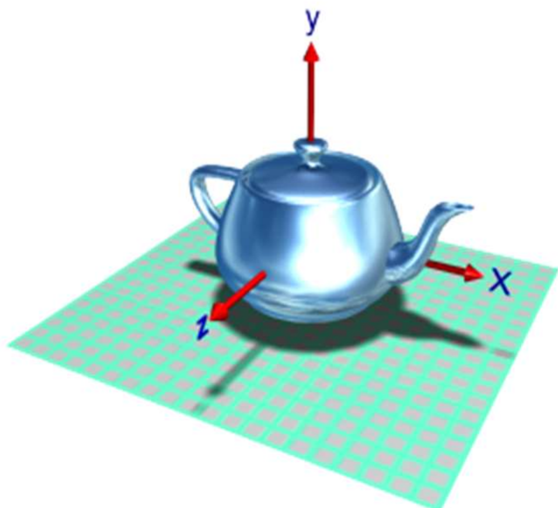


Translate by <0, 1, 0>

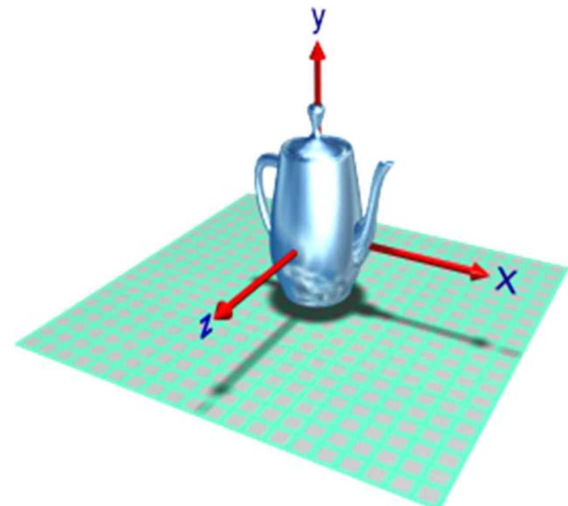
Scale



<1, 1, 1>

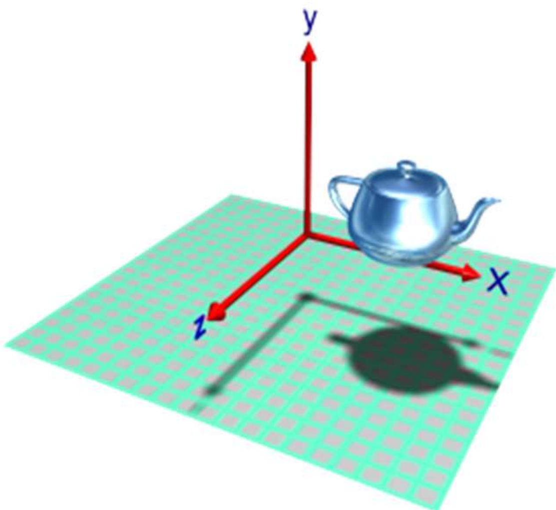


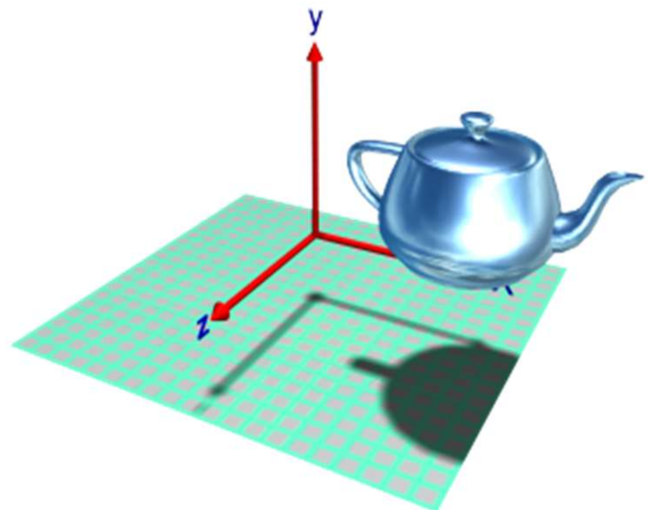
<2, 2, 2>



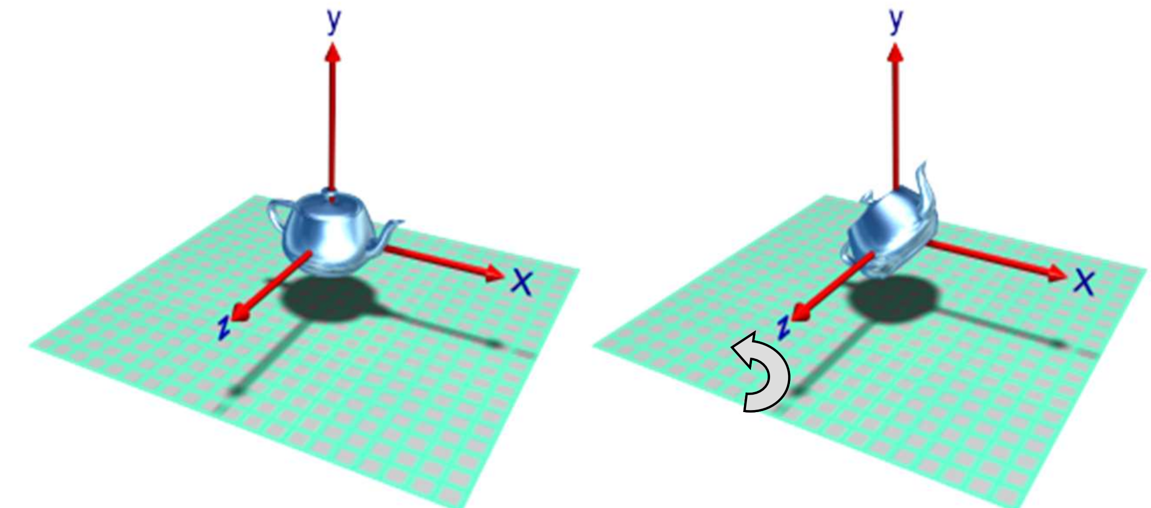
<1, 2, 1> (non-uniform scale)

Please Note: distance from the origin also scales

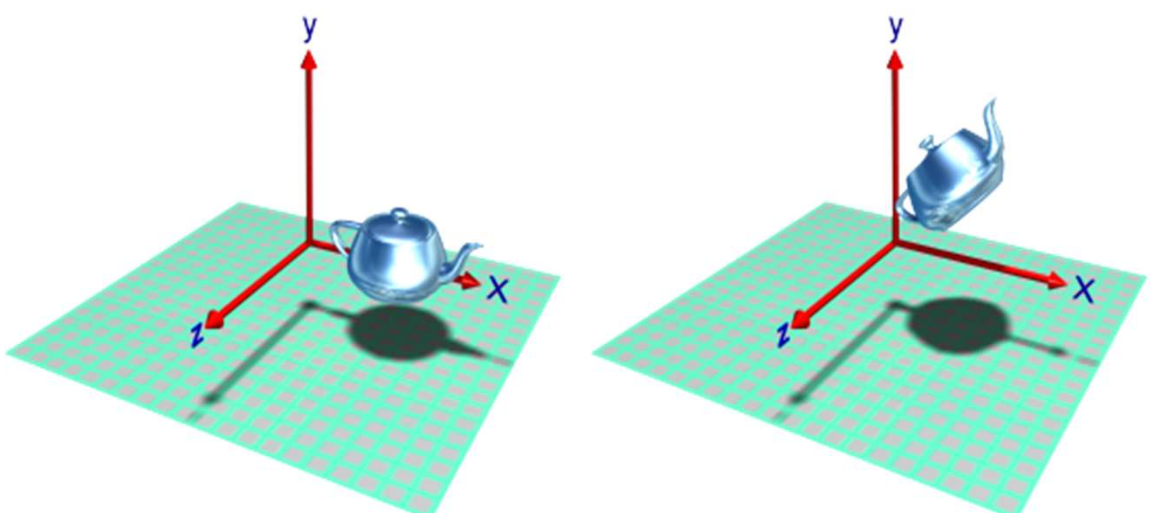




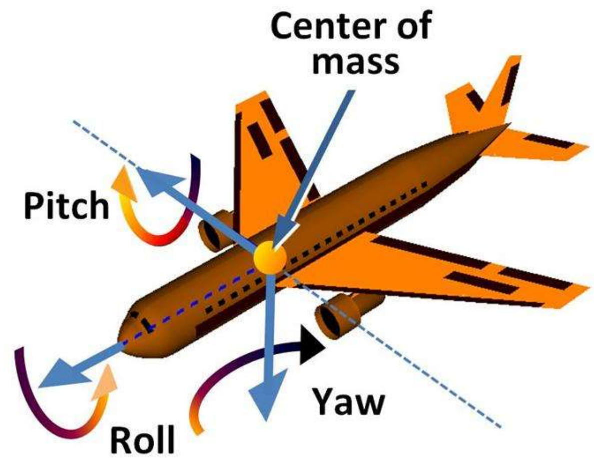
Rotation



rotation of 45^o about the Z axis



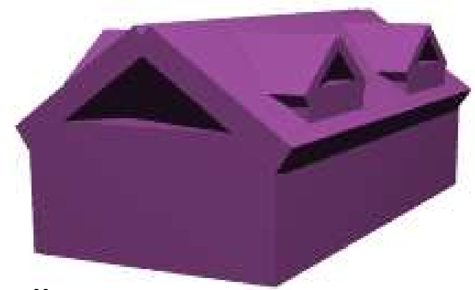
offset from origin rotation

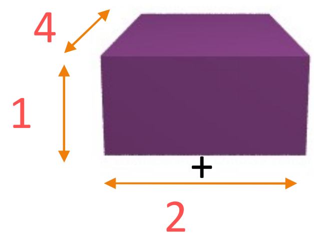


Directional vector describing axis of rotation

Modelling and Transforms

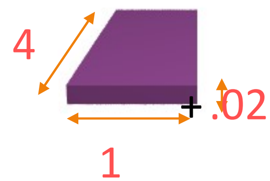
Modelling a house with dormer windows





1. Create a rectangular block as the base of the "house"

2. For the left half of the roof: make a thinner narrower block of the same length

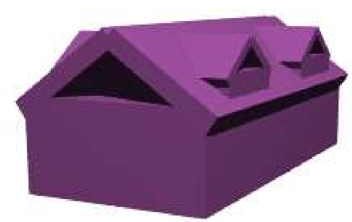


Rotate it by a small angle

3. Repeat for the right half of the roof (rotating the other way)

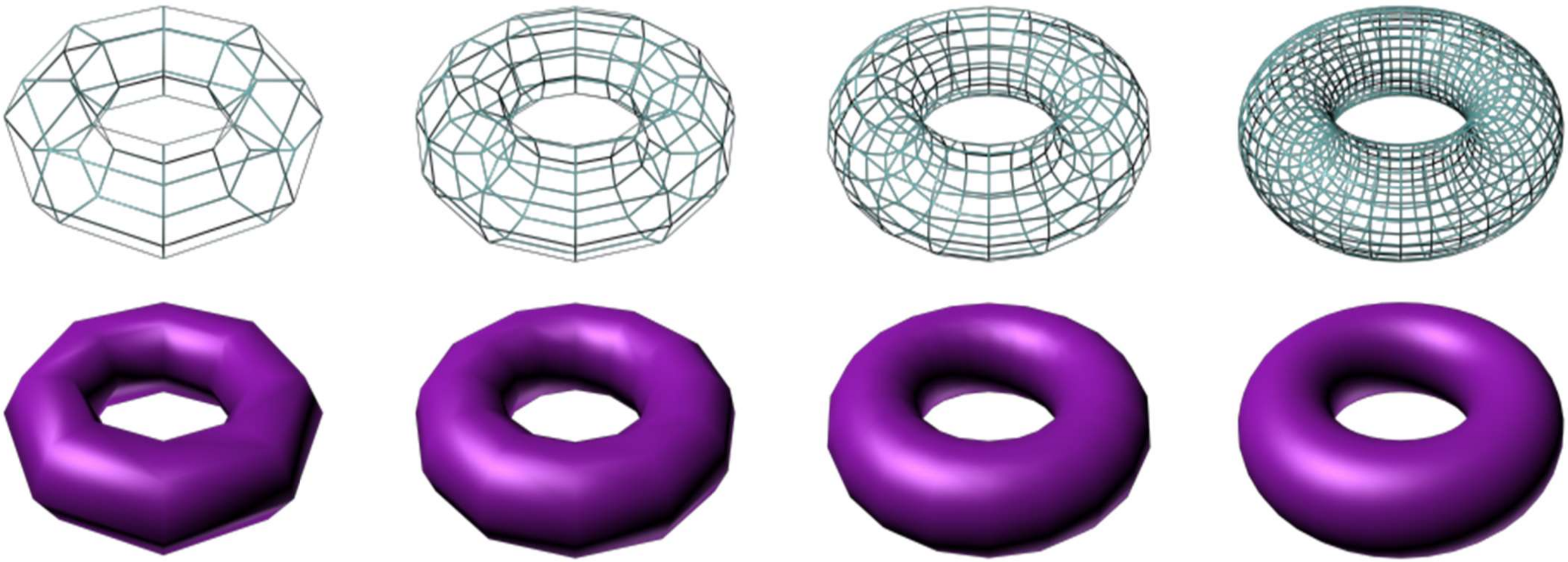
4. Put it all together by **moving** pieces into place

5. Create a copy of the house, and **scale** it down to a fraction of the original. Move this into position to serve as a "dormer window"
Repeat for all windows



Discretization

48 polygons 120 polygons 300 polygons 1000 polygons



In most cases, mathematical models need to be converted to discrete representations before used in scenes. An appropriate ***resolution*** needs to be found.

Subdivision Surfaces

Geri's Game (1997)

DeRose, T., Kass, M. and Truong, T., 1998, July. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (pp. 85-94).

Subdivision Surfaces in Blender :
https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/subdivision_surface.html

Online resources

Pixar’s in a Box – Character Modelling:
https://www.khanacademy.org/computing/pixar/modeling-character

Blocking in Blender:
https://www.youtube.com/watch?v=4xLdisAvjx8&list=PLa1F2ddGya_-UvuAqHAksYnB0qL9yWDO6&index=44

Modelling in Blender: https://www.youtube.com/watch?v=LbQ6_D-wLQs&list=PLa1F2ddGya_-UvuAqHAksYnB0qL9yWDO6&index=45

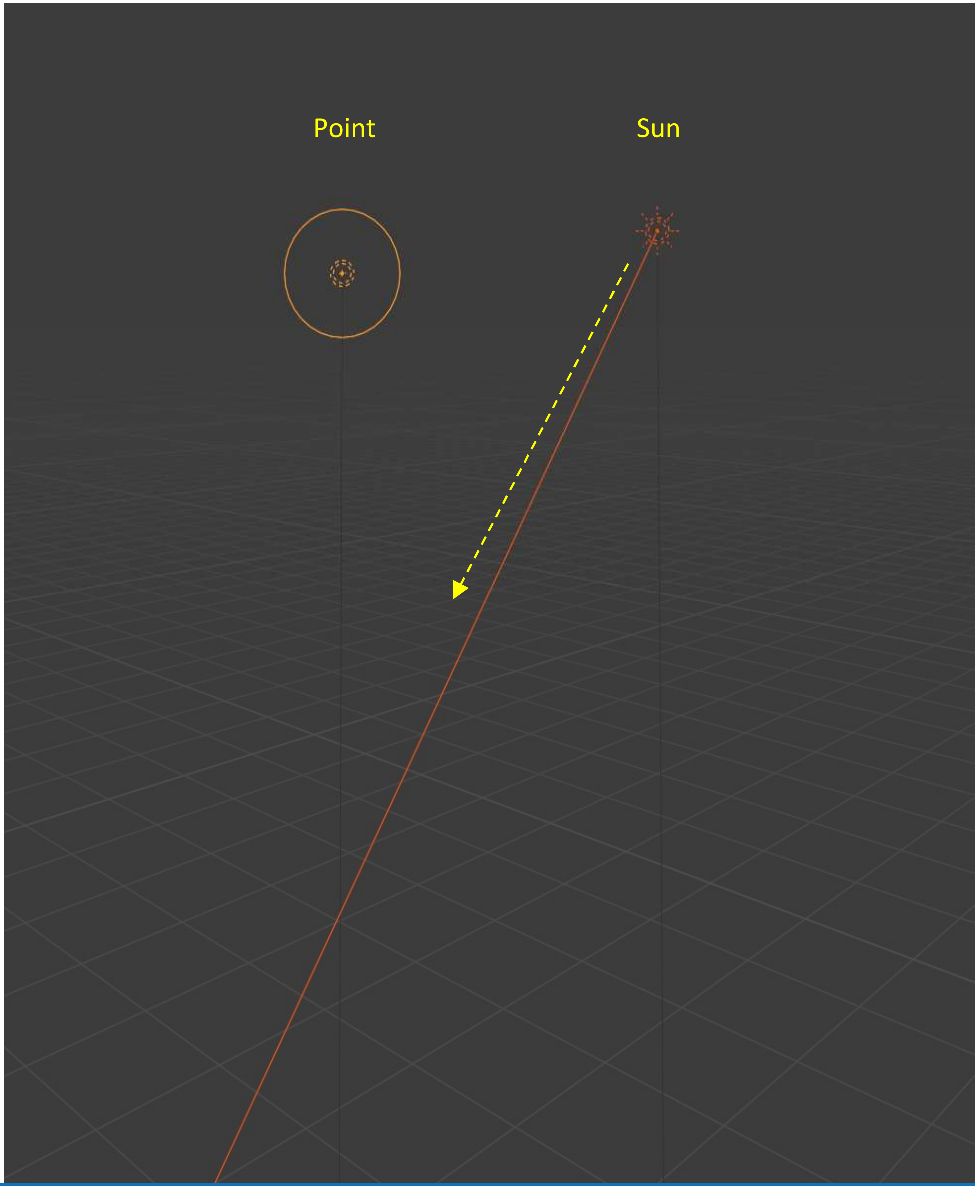
Lights, Materials & Rendering

Image – Anonymous Student

Virtual Lights

Type of Virtual Lights in Blender

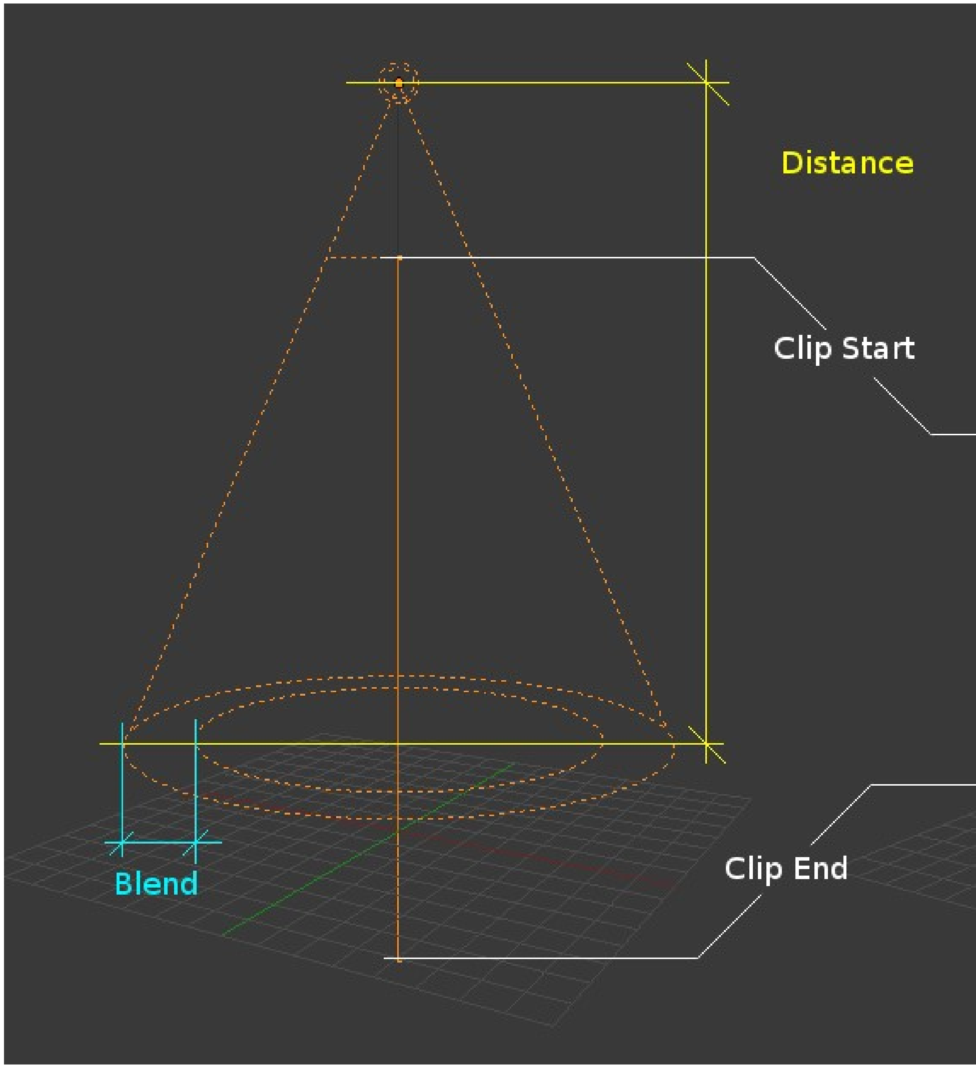
- Point lights
- Sun (directional) lights



Virtual Lights

Type of Virtual Lights in Blender

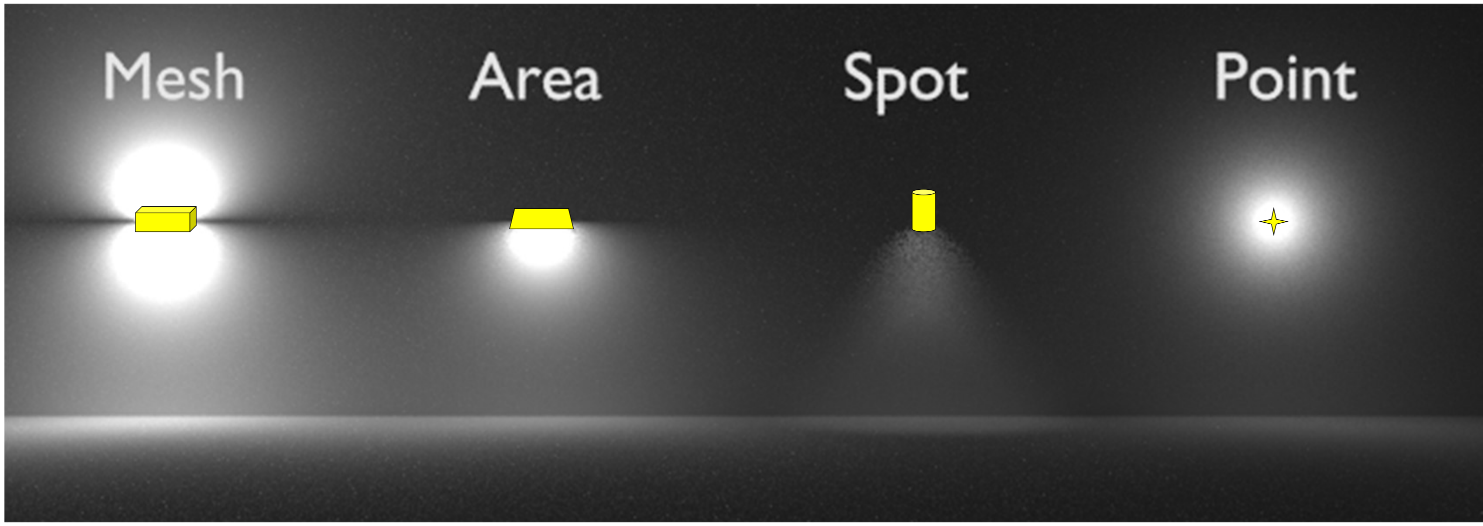
– Spot lights



Virtual Lights

Type of Virtual Lights in Blender

- Area light
- Mesh light



Materials

Shaders

- Describe how the object should be shaded (hence the term *shaders*)
- Early shaders only describe the surface, e.g. Phong shader consisting of *ambient, diffuse* and *specular* components

Phong without spec.

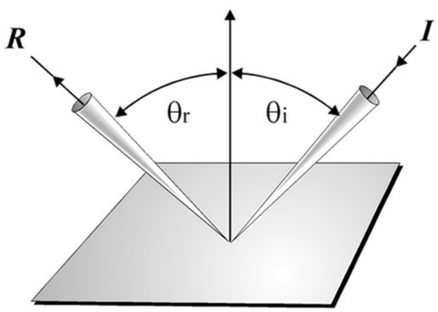
Phong with spec.

Materials

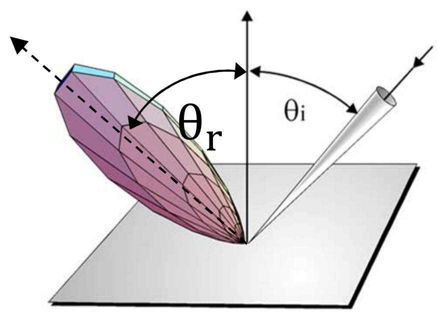
Materials

– Modern shaders describe how light reflects from the surface, as well as refracts or scatters through inside the object (hence the term *material*).

– For reflection, *Bi-direction Reflectance Distribution Function (BRDF)* the current widely used model.



Phong's perfect spec. reflection

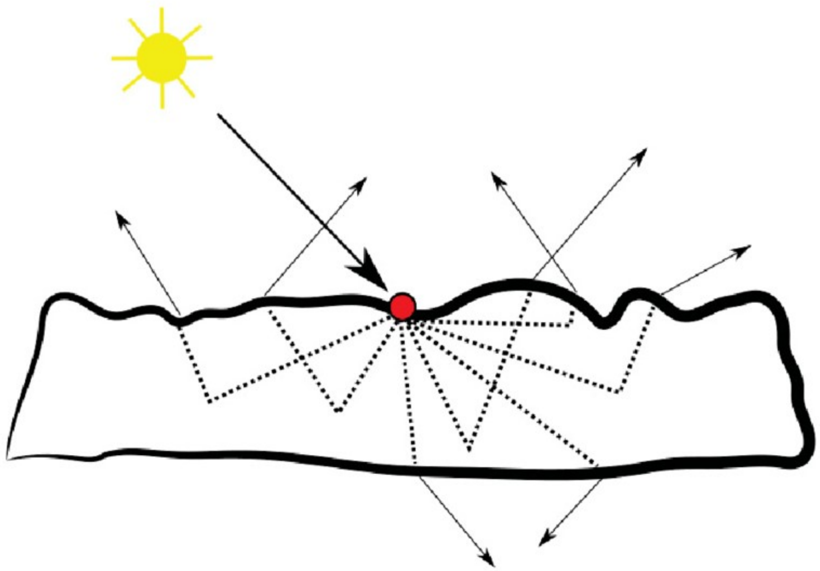


BRDF reflection

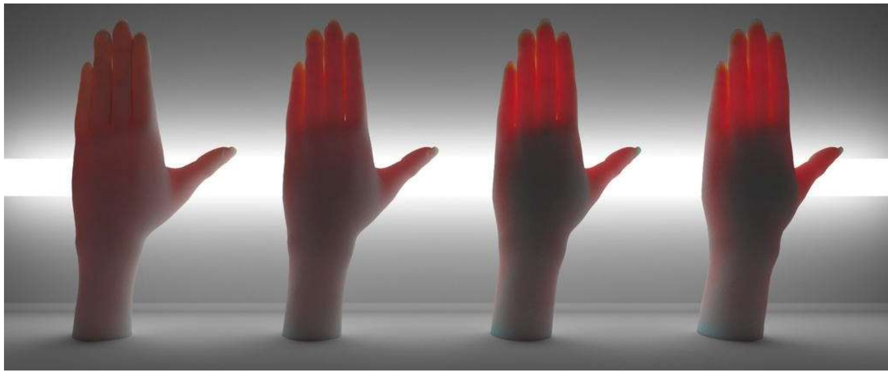
Materials

Advanced Materials

- *Subsurface Scattering (SS)* is another popular type of material used for skin.
- Advanced models are close to real-world behaviors (photorealistic) but more-time consuming to compute.



Subsurface scattering diagram by Banafsheh Azari

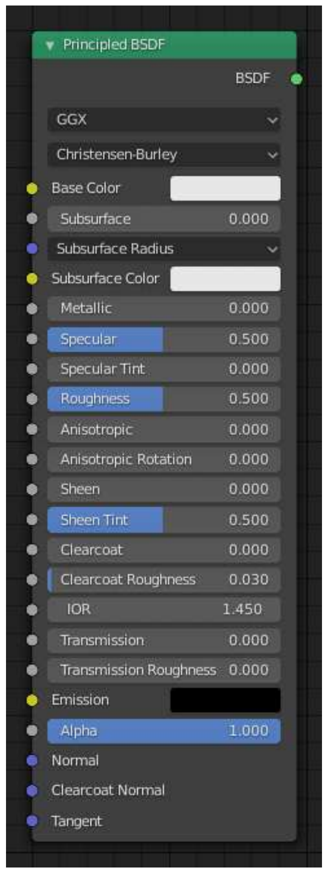


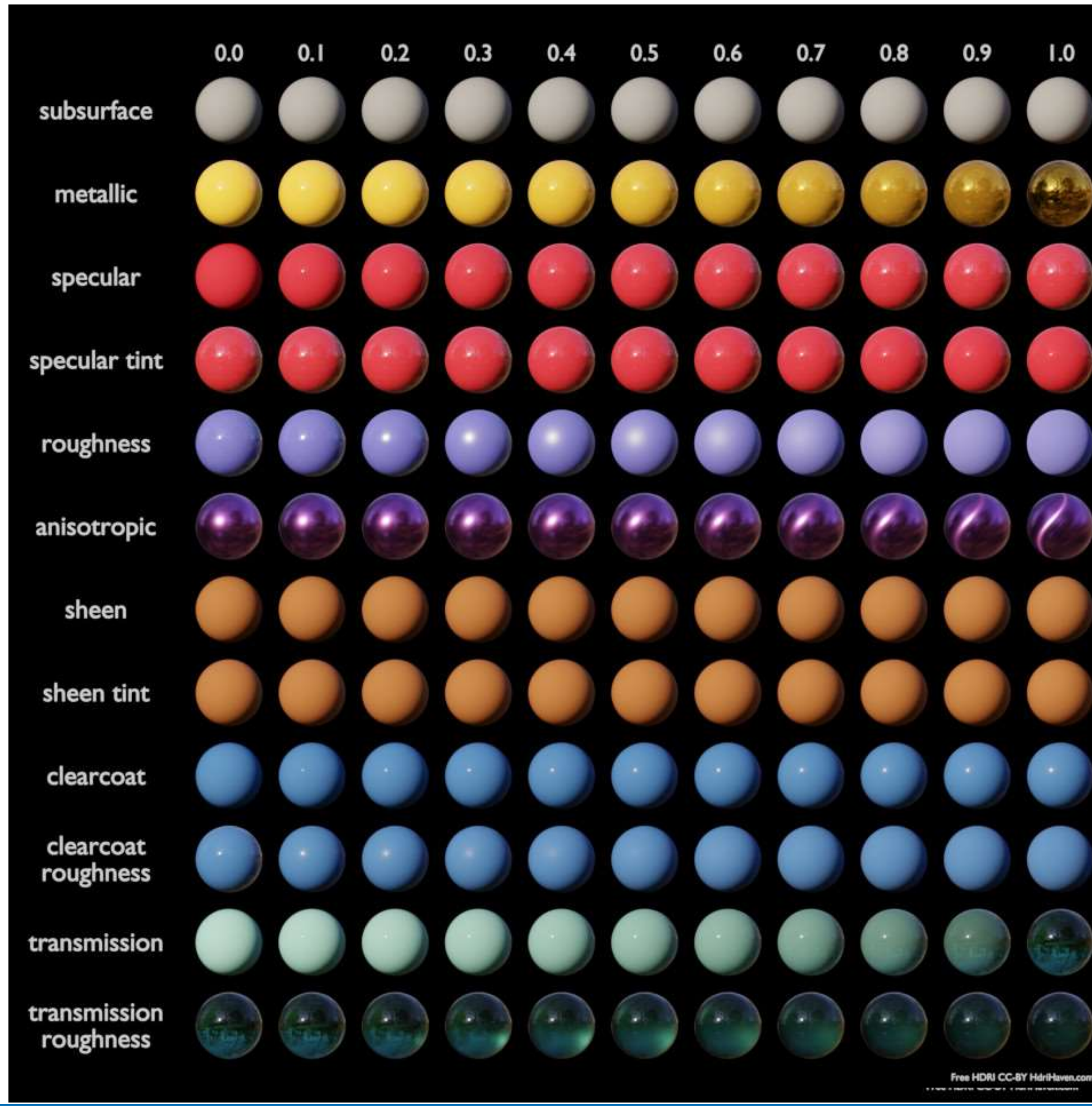
Different levels of SS in Blender created by anul147 on DeviantArt

Materials

Blender Principled BSDF Material

– *Bidirectional Scattering Distribution Function (BRDF + SS)*

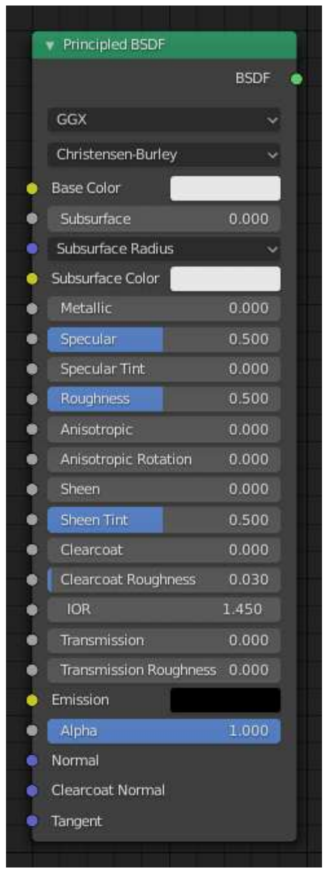


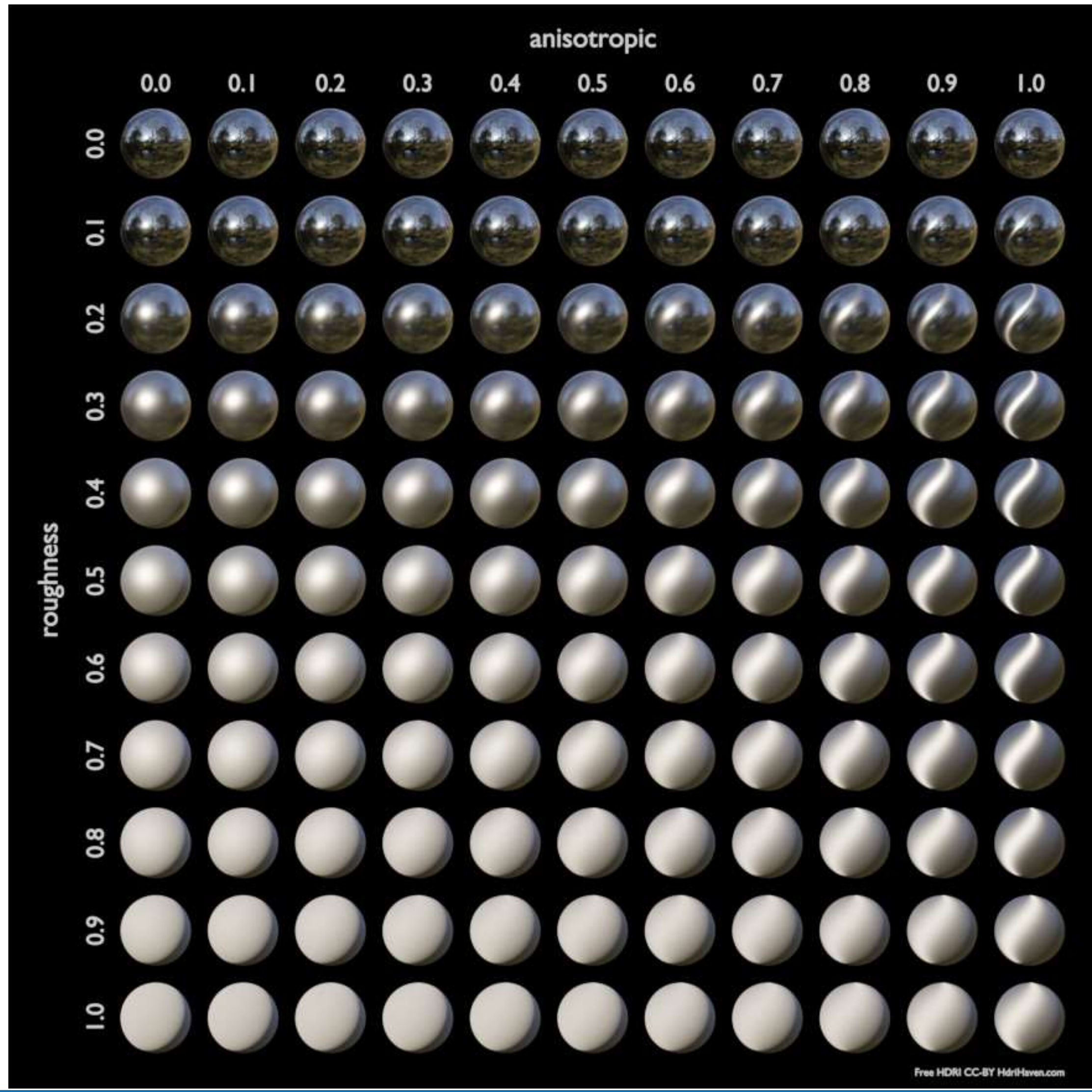


Materials

Blender Principled BSDF Material

– *Bidirectional Scattering Distribution Function (BRDF + SS)*

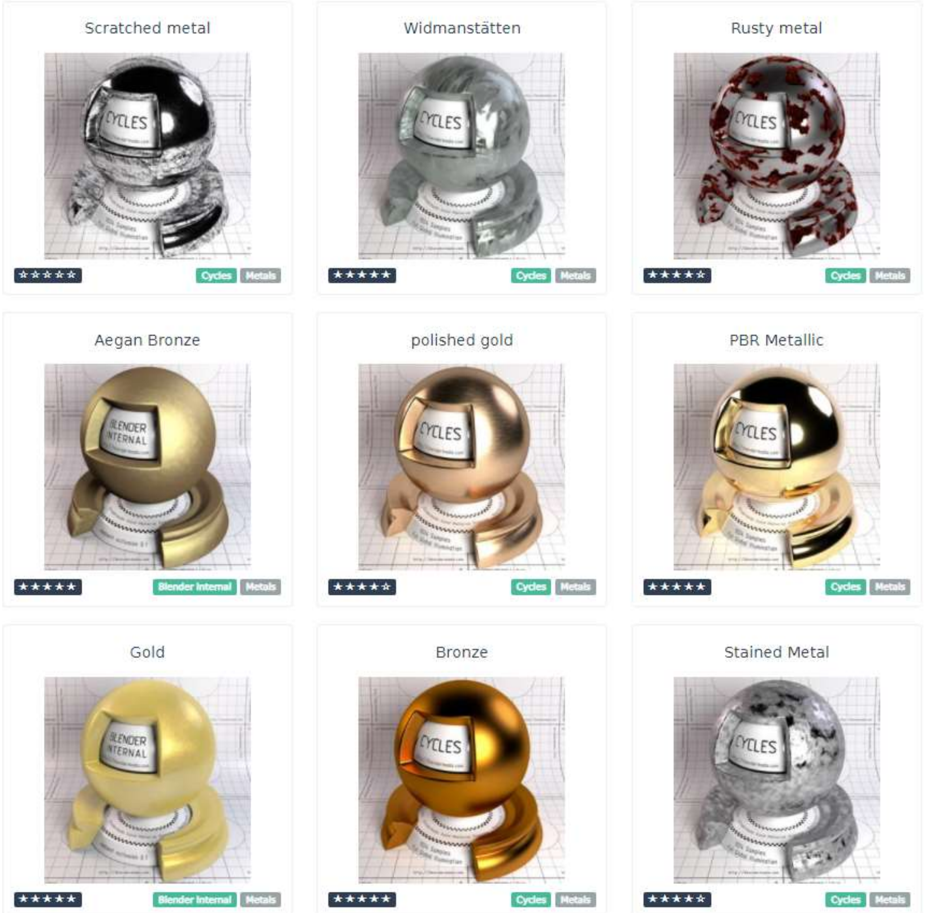




Materials

Free material libraries

- *cc0textures.com*
- *blenderkit.com*
- *cgbookcase.com*
- *poliigon.com*
- *texturehaven.com*
- *sharetextures.com*
- *3dtextures.me*
- *textures.com*
- *blendermada.com*
- *textureninja.com*

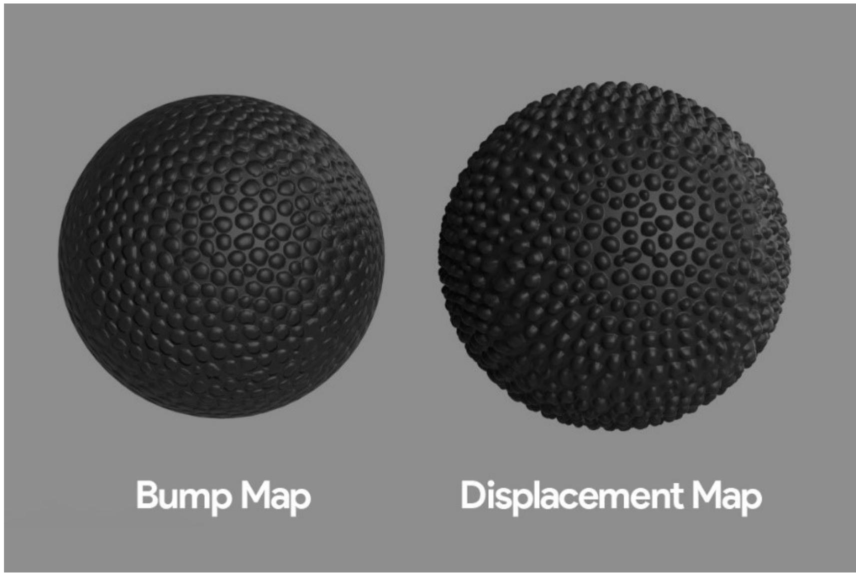


blendermada.com

Materials or Shaders

Materials or Shaders
(surfacing artist, texture artist, look-dev artist)

- Surface
  - color, specular, roughness
  - defined mathematically
    (BSDF and noise patterns)
  - sampled from an image (texture)
- Subsurface
  - Microstructure
  - refraction
  - translucency
- Bump, displacement or normal map
  - Macrostructure (bumps) on the surface



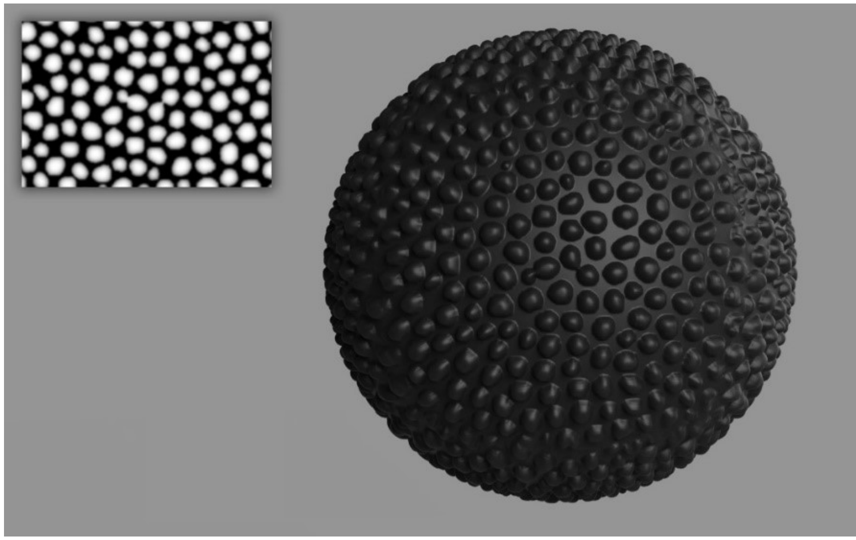


Image courtesy of yakodsign.com

UV & Texture

UV mapping and UV unwrapping

A quick way to add details to a model is flattening the surface around the model and then paint on it.

Think of it as unwrapping a candy wrapper, paint on the wrapper and then rewrap it.

In CG, the wrapper is called a ***UV Map*** *(location info)* and the painted wrapper is called a ***UV Texture*** *(color info).* UV unwrapping is considered an advanced topic and we will not have enough time cover it in this lab.

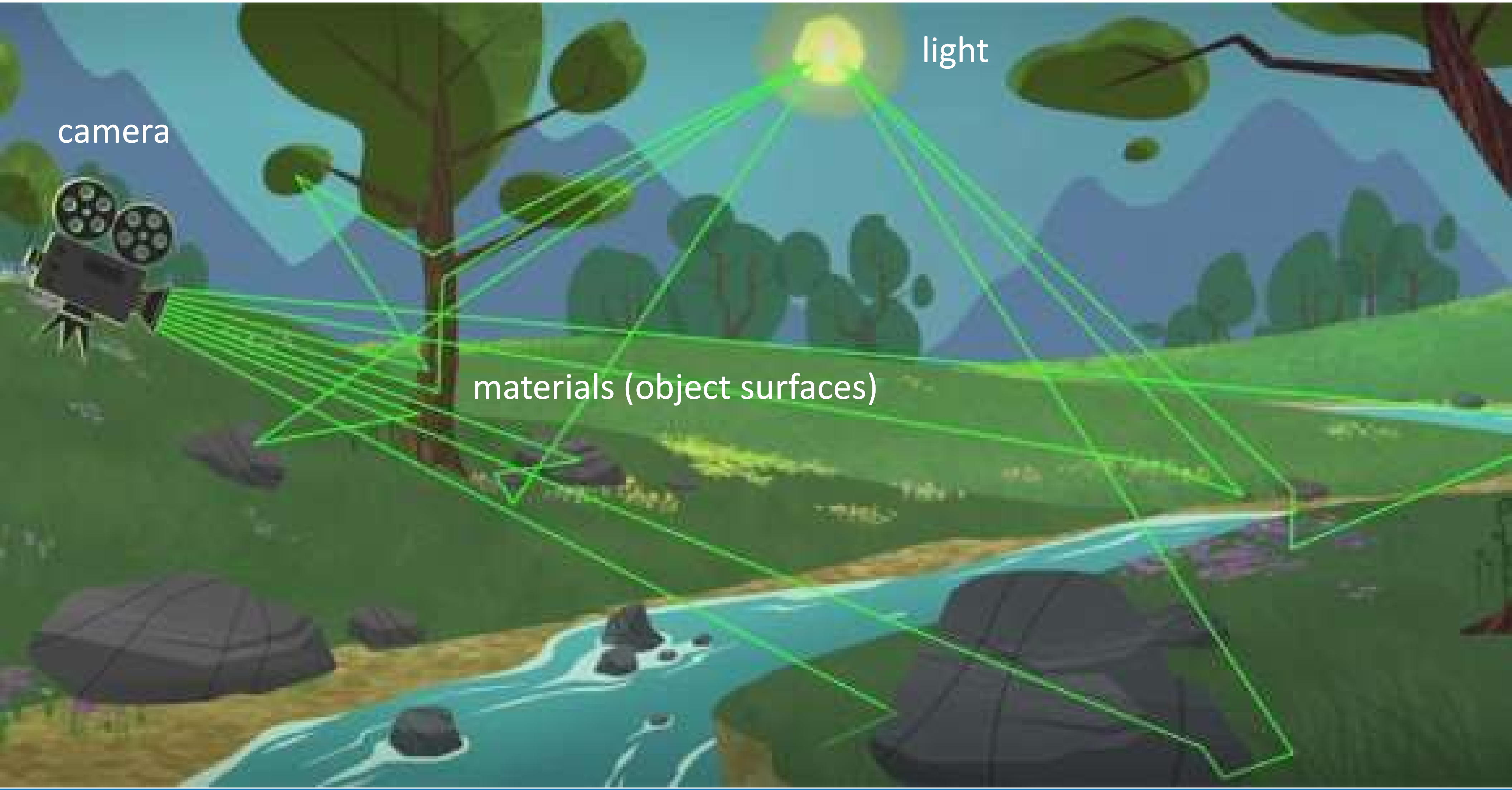
Please watch these two videos in your free time.

- Introduction to UV Mapping
- Intermediate UV Unwrapping

Image courtesy of Widhi Muttaqien

Rendering

Disney's Hyperion Renderer



Animation

Animation

Synthesis of motion – creation of "life"

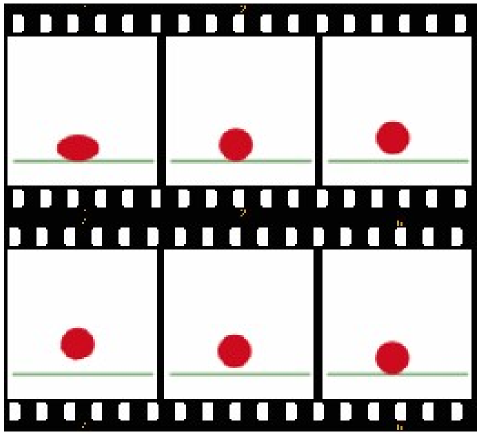
The Illusion of motion created by consecutive display of static elements – through Persistence of Vision

- Eye retains an afterimage for a short amount of time
- Persistence of Vision induces the illusion of continuity

Typical Framerates

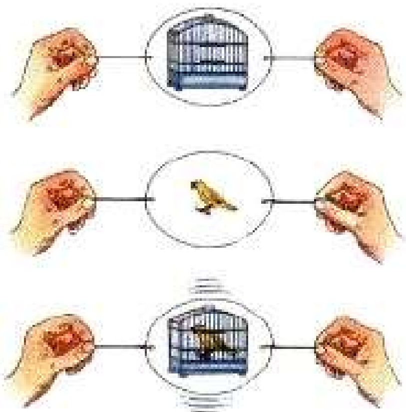
- LCD's 60Hz+
- Video 50/60Hz
- Movie 24Hz
- Cartoons 12Hz

Insufficient frame rates result in flicker



© Branko

Persistence of Vision



Thaumatrope 1824

Phenakistoscope 1832

Zoetrope 1834

Laura Hayes and John Howard Wileman Exhibit of Optical Toys
http://courses.ncssm.edu/gallery/collections/toys/opticaltoys.htm

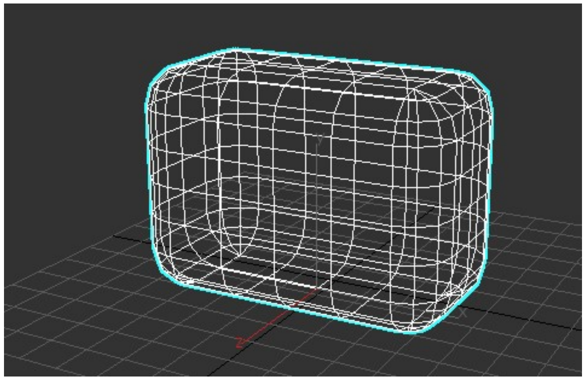
3D Zoetrope

Pixar's 3D Zoetrope at Disneyland California
Video: https://www.youtube.com/watch?v=5khDGKGv088

Modeling vs. Animation

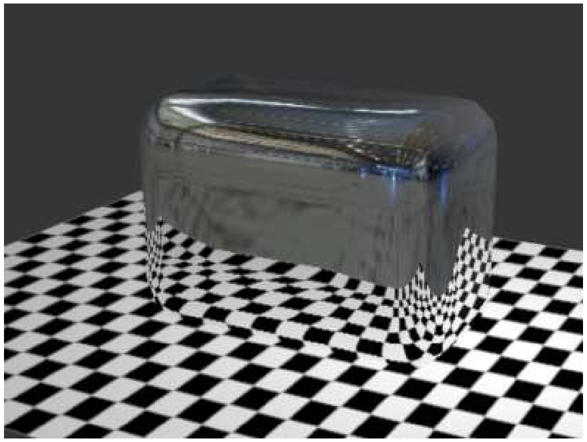
Modelling:

– What are the parameters to define objects?



Rendering:

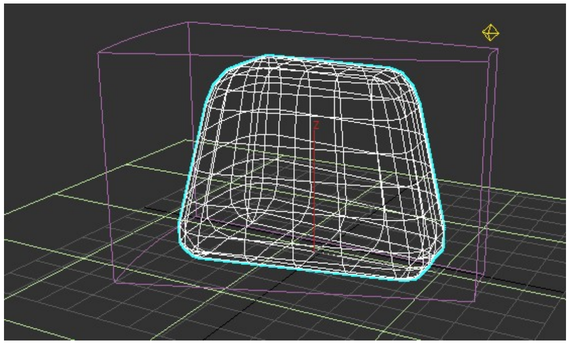
– How to display the model?



Animation:

– How do the parameters change? / How does the model change?

– Motion synthesis



Types of Animation

Scripted
- Keyframes + interpolation

Data-driven
- Performance capture, human input

Procedural
- Synthesised, rule-based, automated, behavioural, heuristics, physically based

Key Frame Animation

KEY frame's drawn by lead artist

In-betweening frames filled in

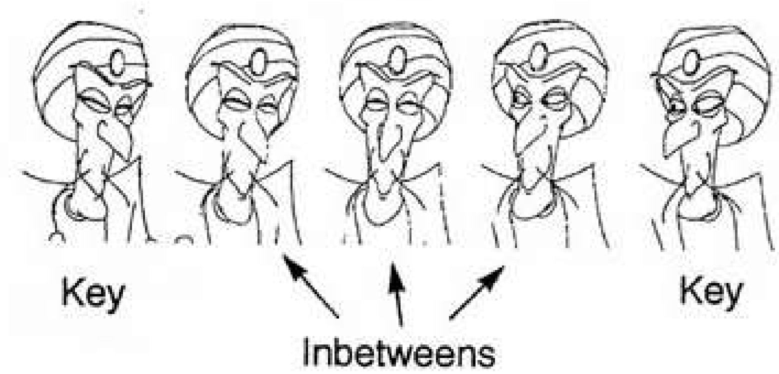
In Computer Animation:

– automatic in-betweening by various **interpolation** methods.

– Due to re-usable data, we can interpolate:

  - Frames / images
  - 3D data/ 3D transforms
  - Other Parameters

Image source: Midori Kitagawa



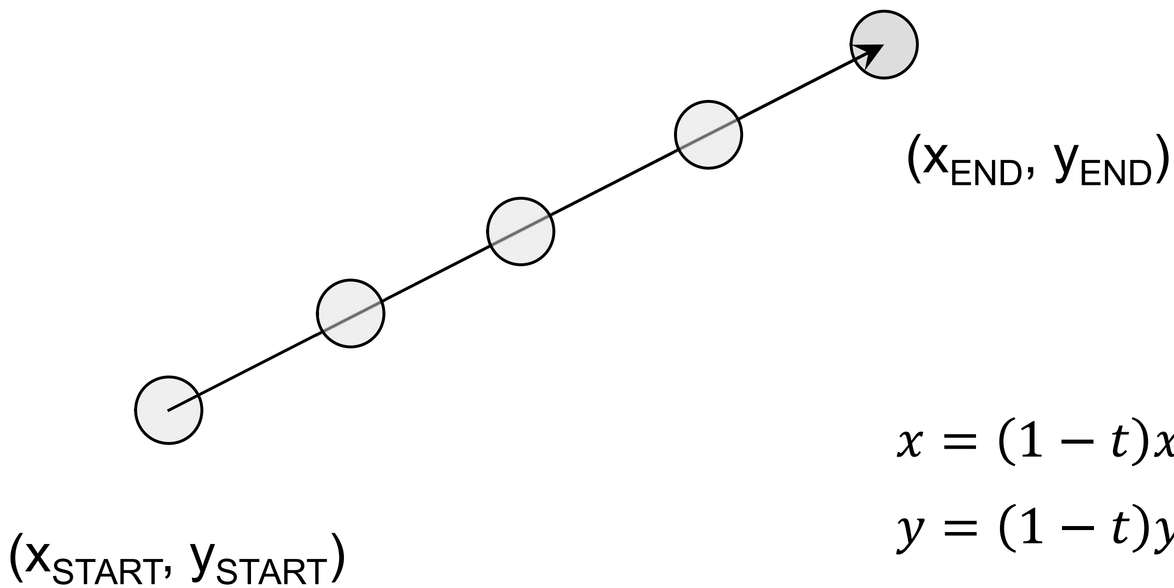
Given known points (e.g. start and end point in a line), INTERPOLATION is the process of mathematically estimating points that lie in between.

Interpolation

In CG Animation, in-betweening is done by interpolation

Interpolation

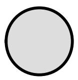
Simplest type of interpolation is Linear Interpolation



$$x = (1-t)x_{START} + tx_{END}$$
$$y = (1-t)y_{START} + ty_{END}$$

Interpolation

Simplest type of interpolation is Linear Interpolation

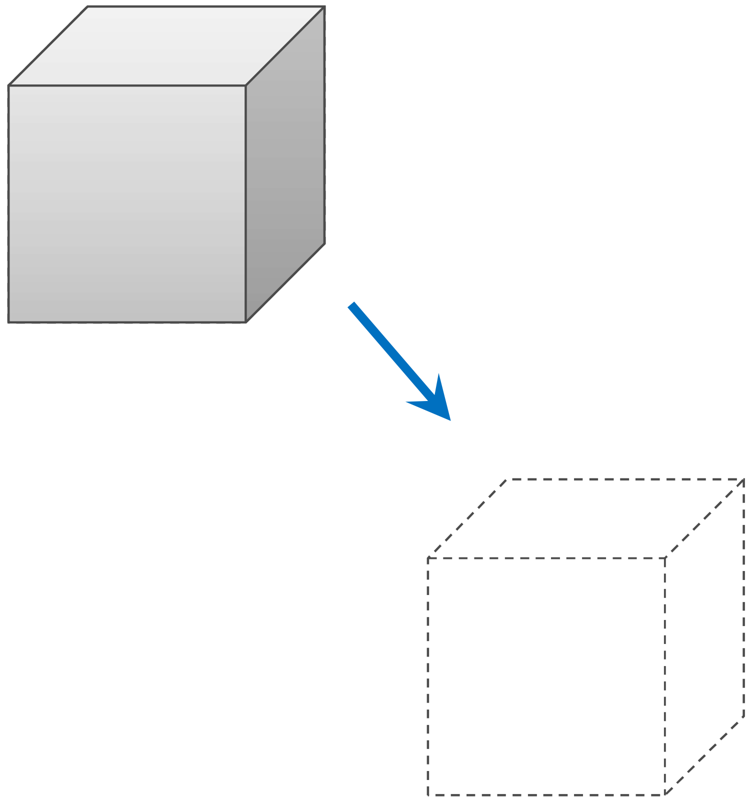


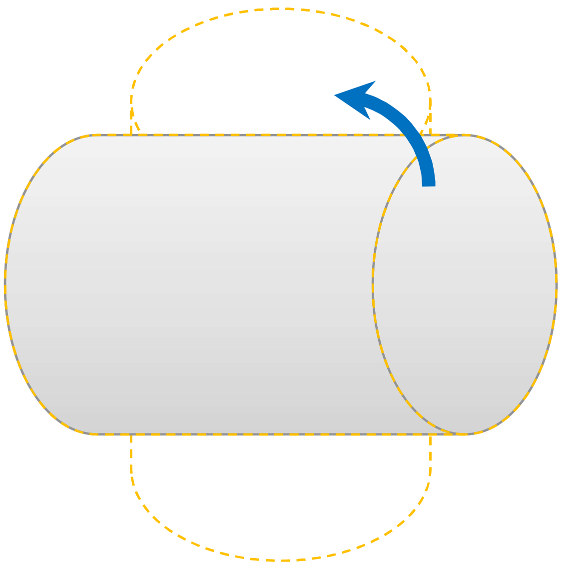
$$x = (1 - t)x_{START} + tx_{END}$$

$$y = (1 - t)y_{START} + ty_{END}$$

Rigid Body Animation

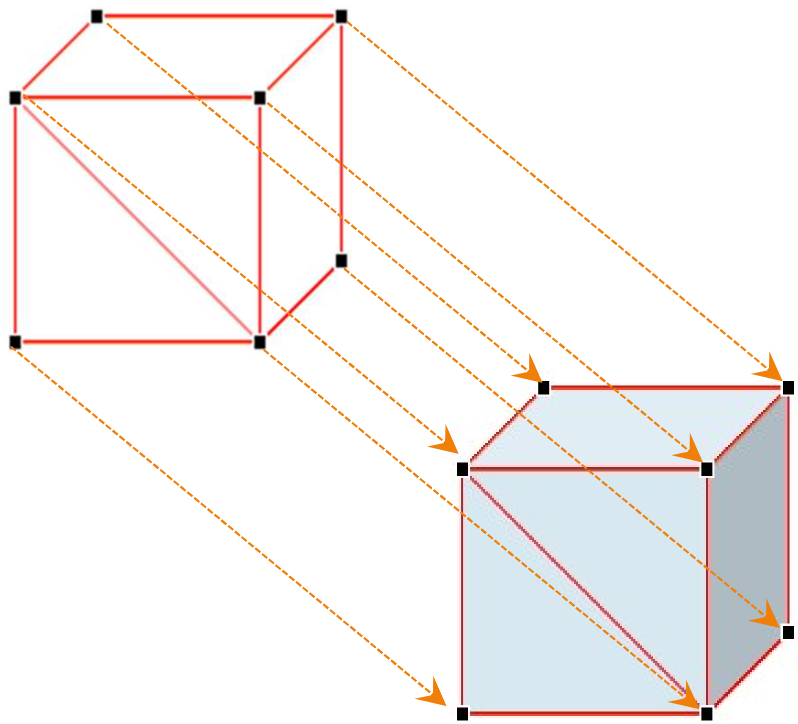
Translations, rotations applied to geometric models… in 3D

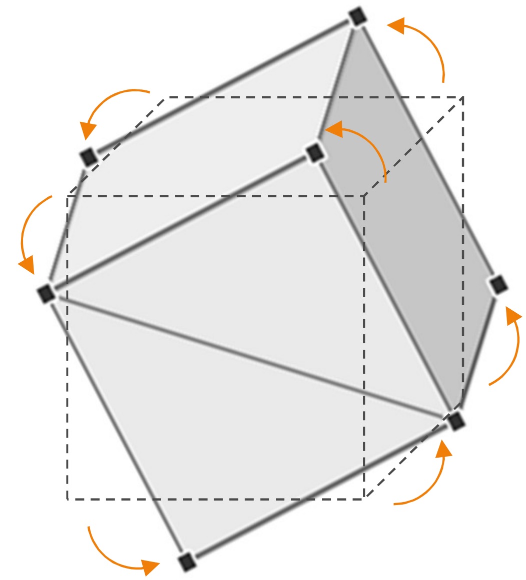




Rigid Body Animation

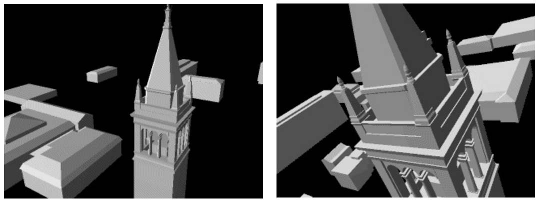
Translations, rotations applied to geometric models… in 3D



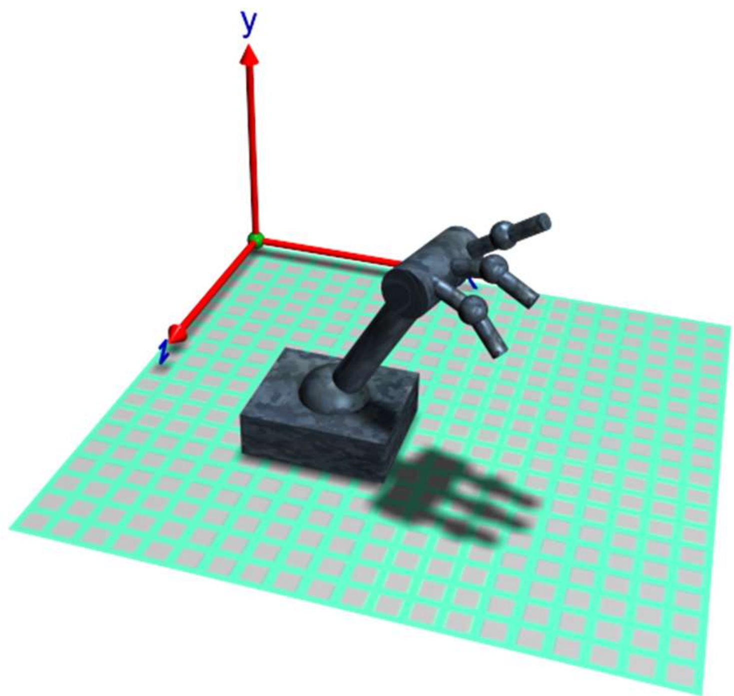


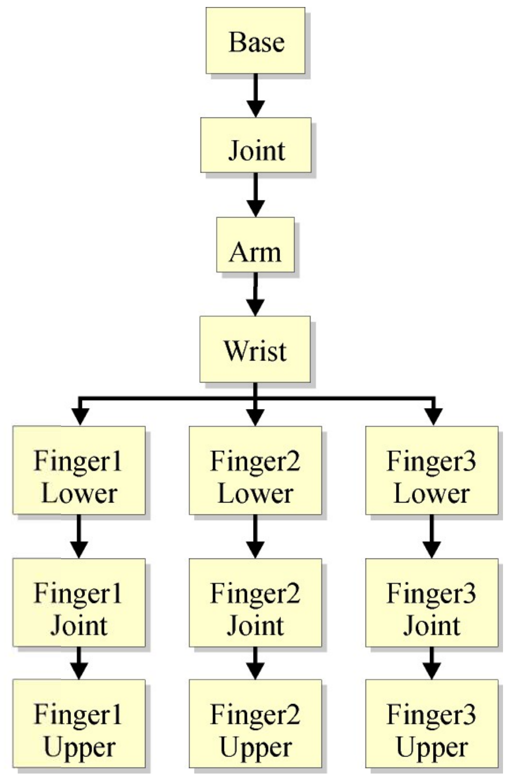
Essentially, each point is transformed (move, rotate, etc.) by the same operation by a discrete amount for each in-between frame.

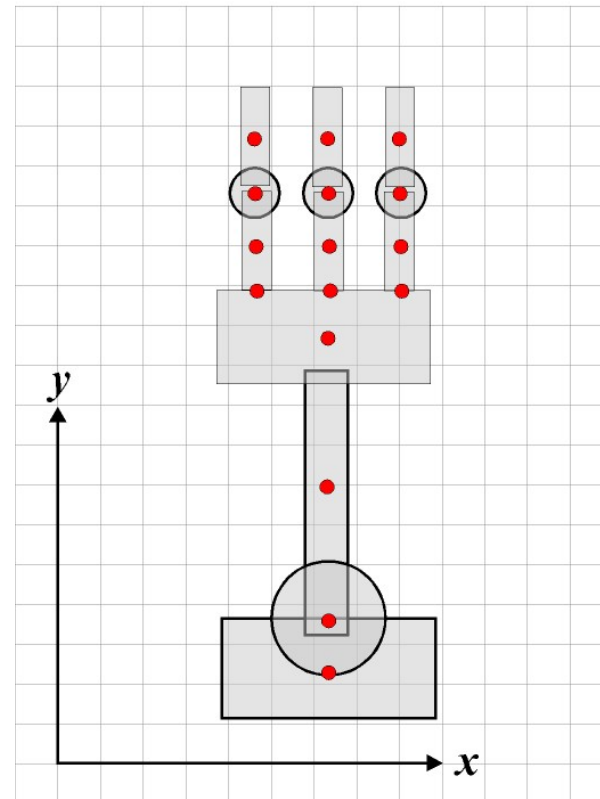
Can also apply to Lights, Camera etc.



Articulated Structures





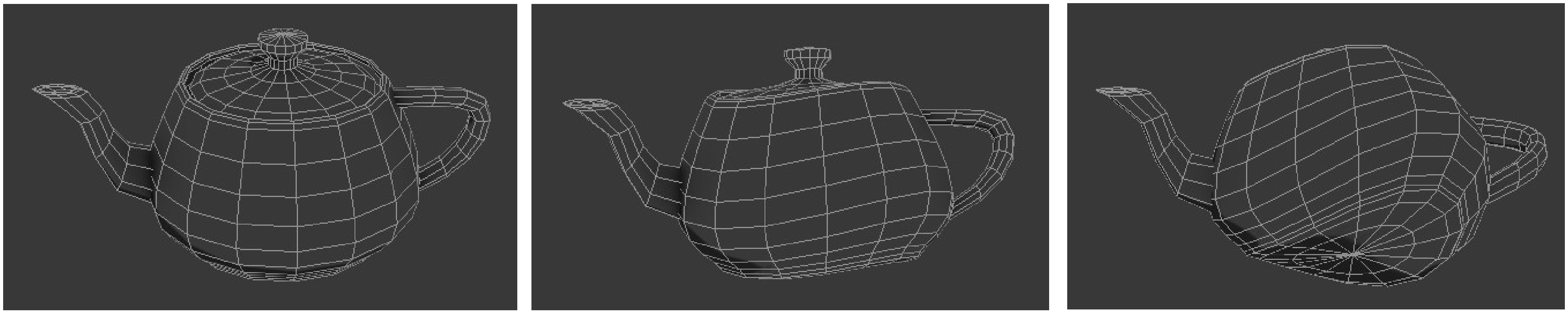


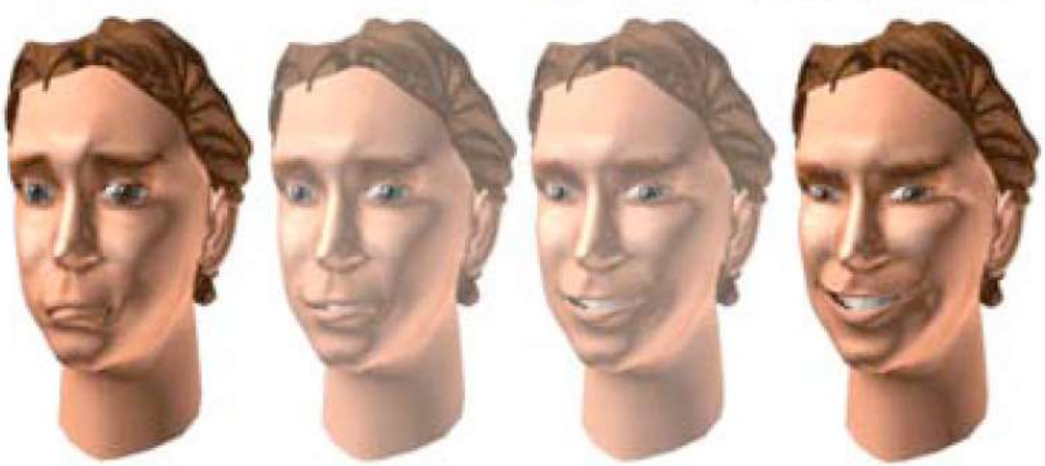
Rigid Hierarchies

Each *"part"* is a rigid transform (all points are transformed by same amount)

Each part has it's *"parent's"* transforms plus it's own

Deformation and interpolation





Points on the mesh have different transforms (move differently relative to each other). However each point is still interpolated between start and end *pose*.

Motion capture

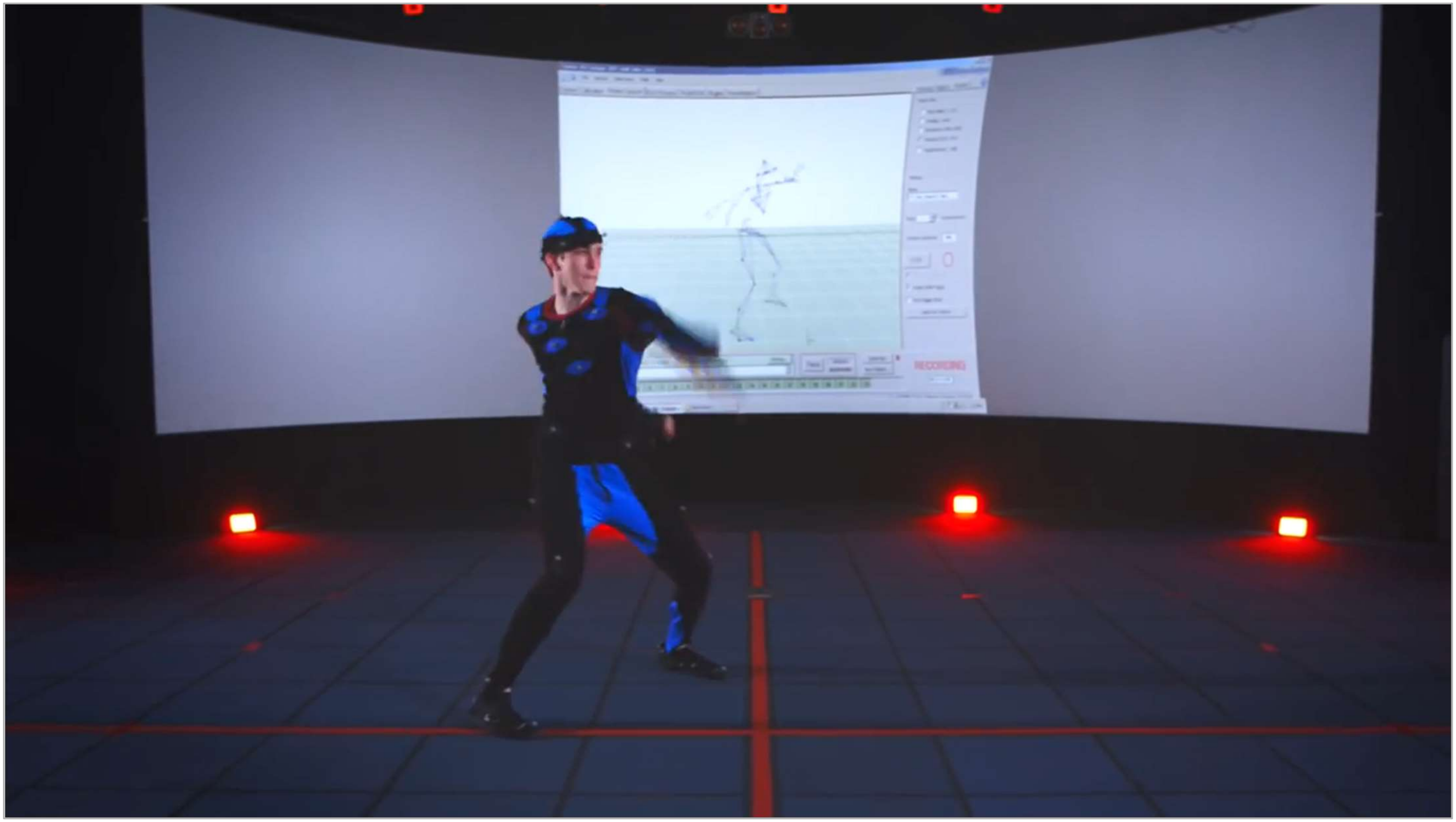
© Columbia Pictures

© Gameloft

© Kinectic.net

See: A Brief History of Motion Capture for Computer Character Animation
http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/motion_capture/history1.htm

Motion capture



Excerpt from “What is Motion Capture” © Full Sail University https://www.youtube.com/watch?v=fm-A1lknrxE

Skinning & RIGGING

Underlying "skeletal" mesh is rigid. Skin moves with bones.

Image © Wolfire Blog

SKINNING & RIGGING

The process of setting up a skeleton to influence the deformation of a mesh is called "rigging."

Image © Wolfire Blog

Procedural Animation

“Mathematical” Models of Motion

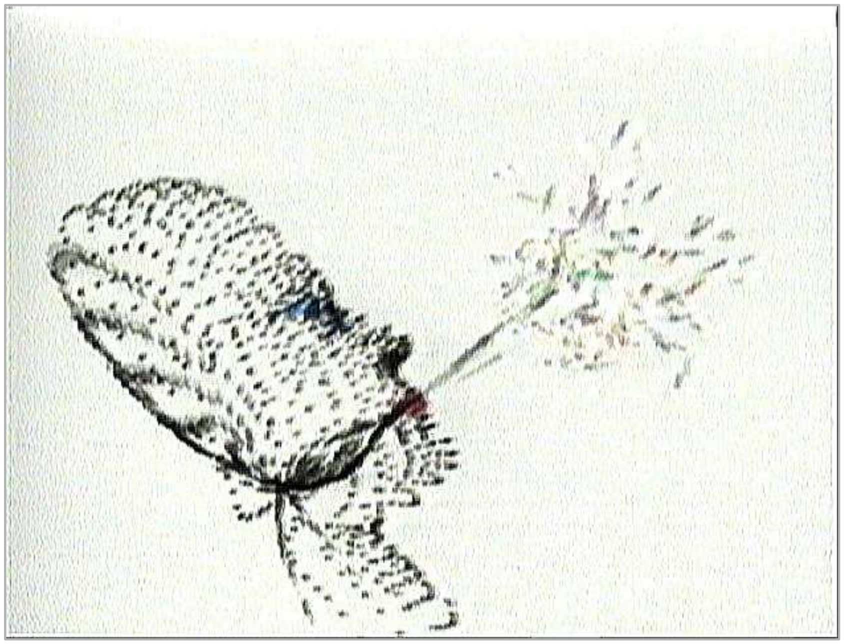
© Yasuo Ohba

© Animusic

The change in position and orientation of an object (or parts of an object) determined by a set of functions/rules.

Particle Animation

Simple rules applied to very simple individual objects



Particle Dreams 1988 © Karl Sims

Collectively can result in useful emergent behaviours

SAMPLE CODE:

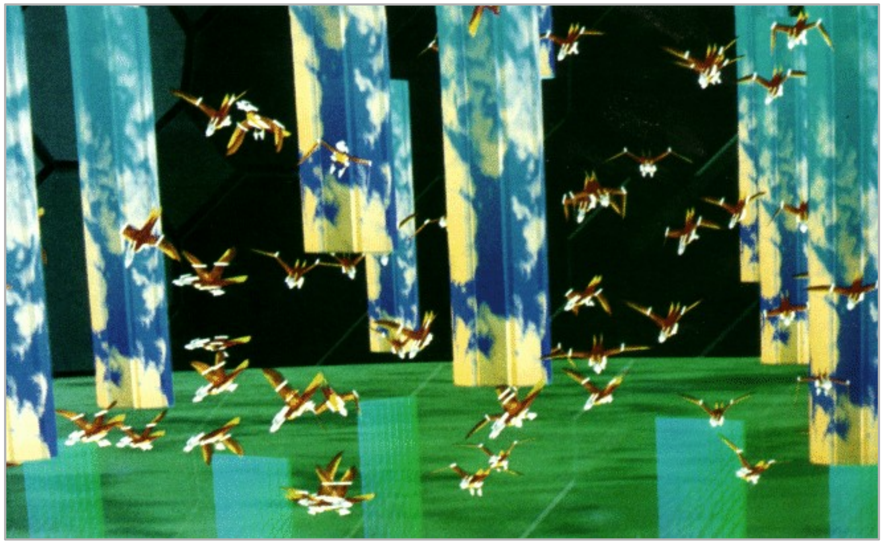
```
Let <px, py> be position of the particle
<vx, vy> be velocity of the particle

Animation Loop (do this every frame):
    Update position using velocity
    px = px + velocityx
    py = py + velocityy

    Update using velocity using gravity
    vx = vx;
    vy = vy - 9.81;

    Collisions (if object falls through the
    floor, bounce it back up)
    if (py < 0)
    vy = vy * -1;
```

The Genesis Effect - Star Trek II: The Wrath of Khan (1983)

Behavioural Animation

Semi-intelligent rules given to control the motion of "agents" e.g

- Run away from lions
- Avoid obstacles
- Avoid other wildebeest

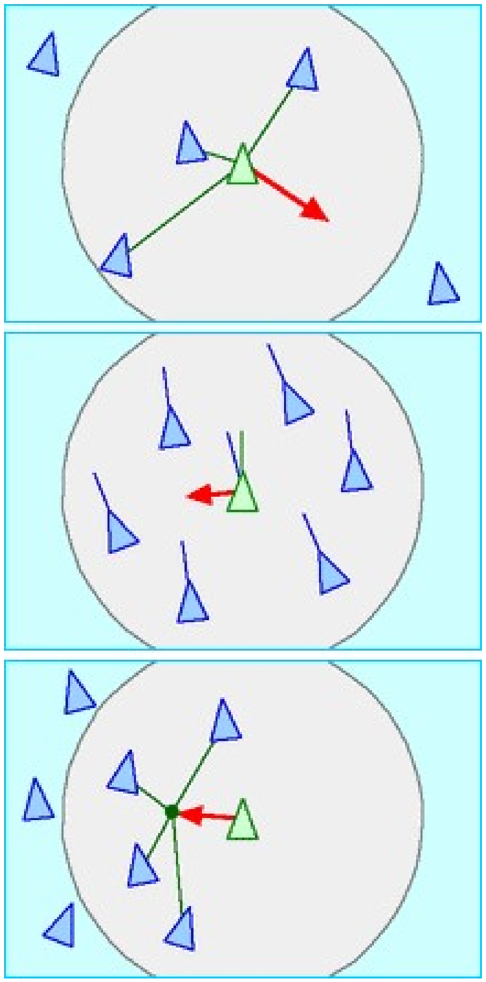


© Reynolds, 1987

Bird-oids

- Useful for modelling animal behaviour
- Flocks, herds and so on

"Boids"



© Reynolds

Each boid maneuvers based on the positions and velocities of nearby flock-mates

Three basic types of steering behaviours

– Separation: avoid crowding

– Alignment: steer towards average heading

– Cohesion: move towards average position

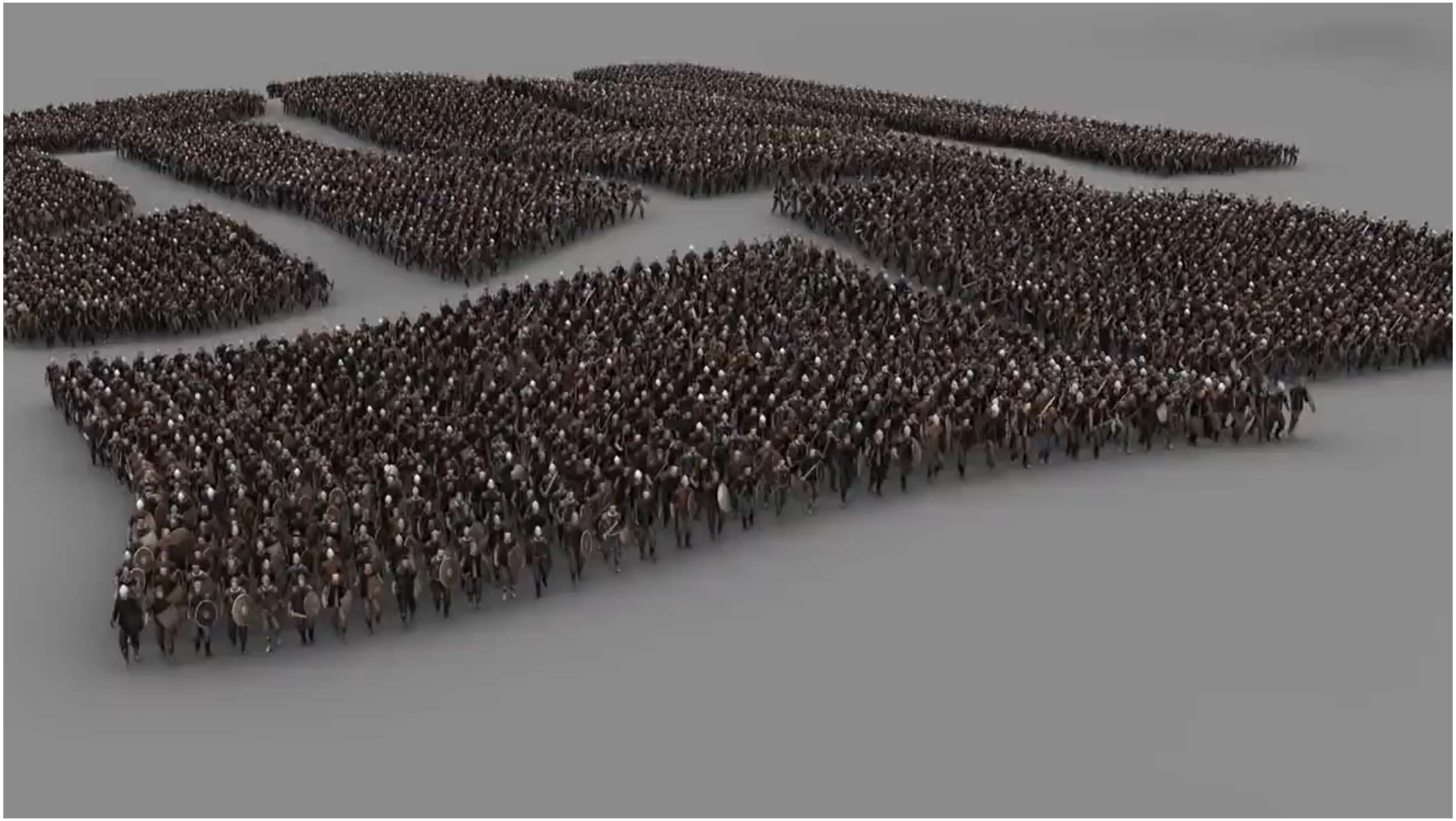
Each boid 'senses' only within local spherical neighbourhood of itself

Boids - Flocking

https://vimeo.com/25025327 © C. Schnellhammer

Crowd Simulation

- Walk towards enemy
- Stay in formation
- Stay safe distance from friends

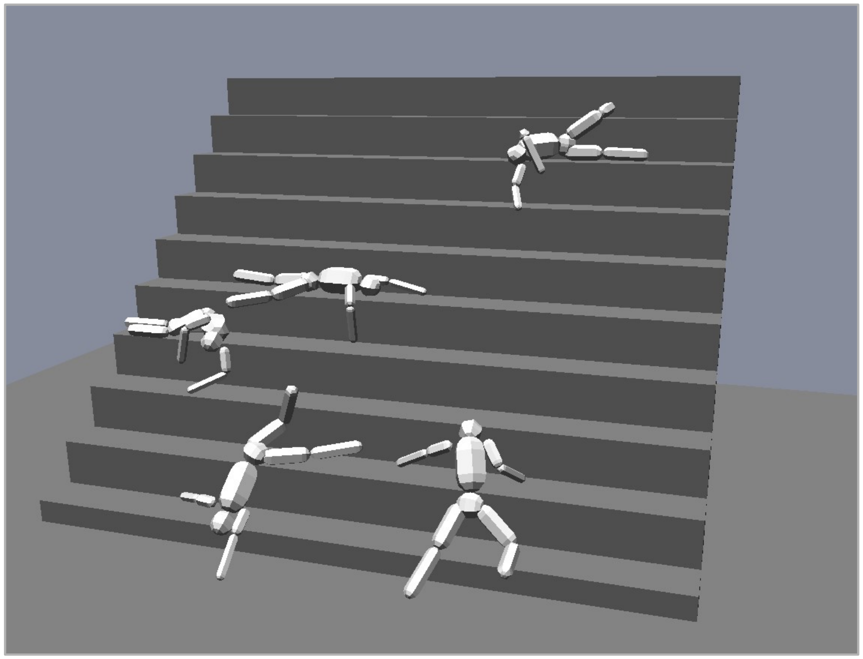


https://www.youtube.com/watch?v=6SqR0dyJ0OQ Massive Crowd Simulation Example, by FX Hive

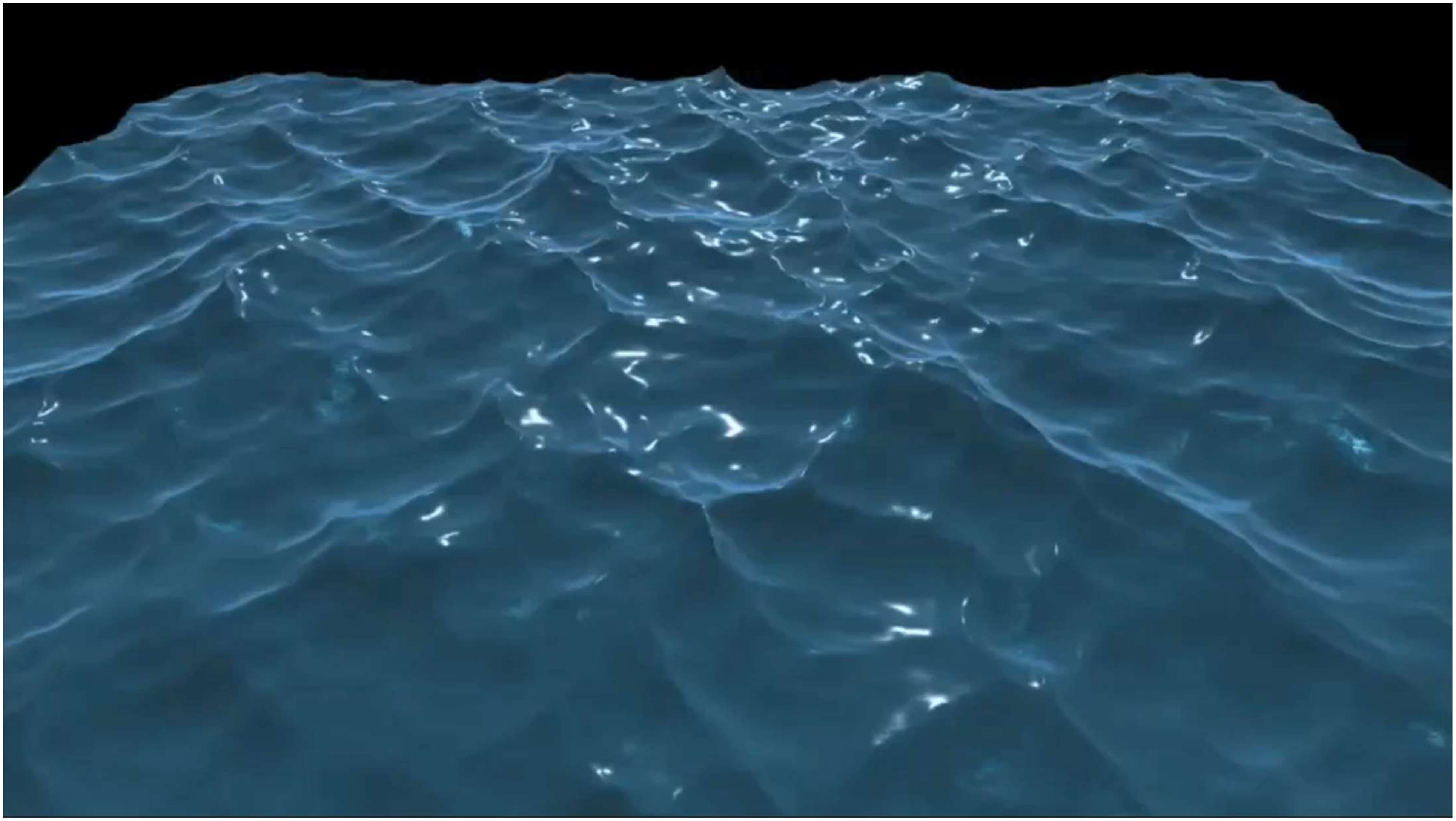
Physically-based Animation

Rules of <u>Physics</u> used to calculate motion

- Collisions
- Forces
  - Friction
  - Gravity
  - Buoyancy
  - Elasticity



Physical Phenomena



https://vimeo.com/16115652 © C. Schnellhammer