Lecture 3: 3D scene overview

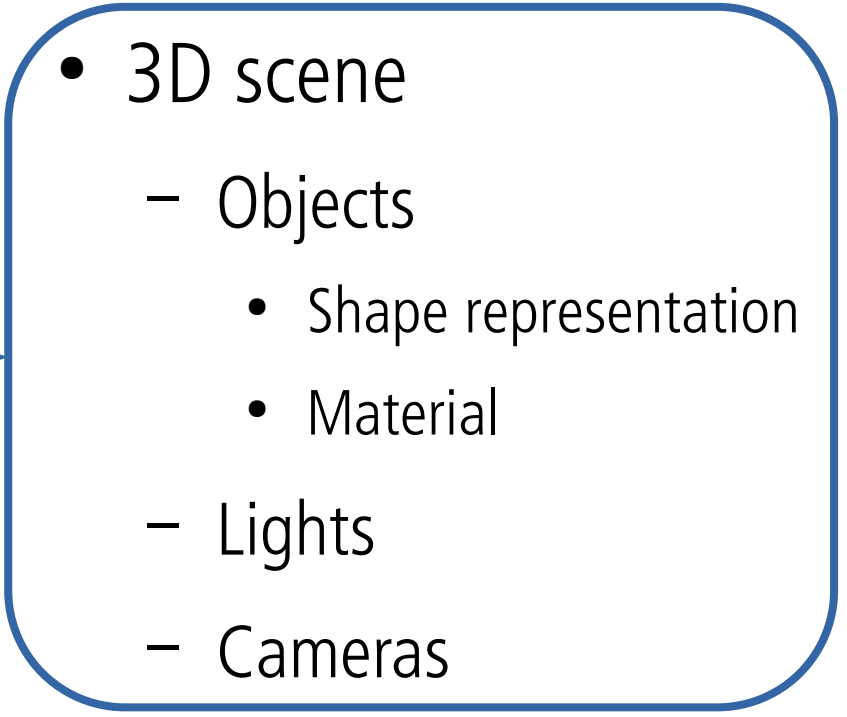
DHBW, Computer Graphics

Lovro Bosnar

1.2.2023.

Syllabus

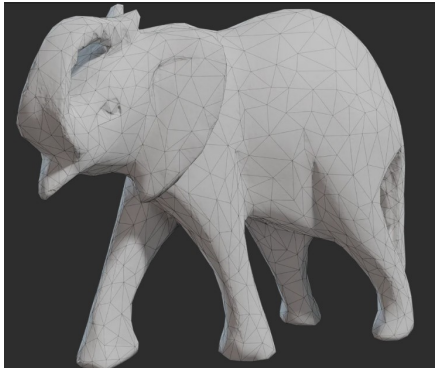
- 3D scene
- Rendering
- Image



3D scene in big picture

Cornerstones of image generation:

- **3D scene modeling**
  - Representing 3D objects, lights and cameras
  - Animation and interaction
- Rendering
  - Creating 2D images from 3D scenes
- Raster image
  - Displaying 2D images
  - Post-processing



https://polyhaven.com/

3D scene modeling for image synthesis

- Image synthesis: rendering a 3D scene
  - Analogy: taking real world photograph/video
- Elements:
  - **Camera** placed somewhere in space
  - Space contains **objects** with various shapes and materials
  - **Light source** emits light in space

Models from:
https://polyhaven.com/

3D scene modeling questions

How do we represent scene elements in a computer?

- Intuitive for user?
- Efficient for rendering?

How we create/acquire/manipulate scene elements?

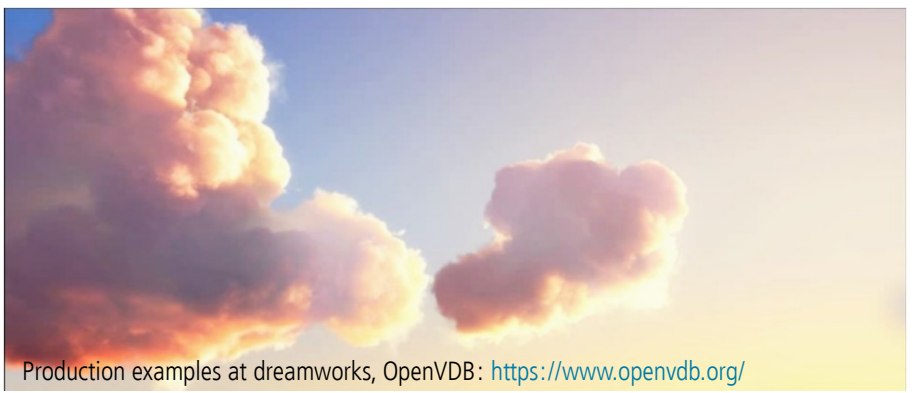
- Objects? Lights? Cameras?
- How to use scene elements to create real-world phenomena and objects?

Objects

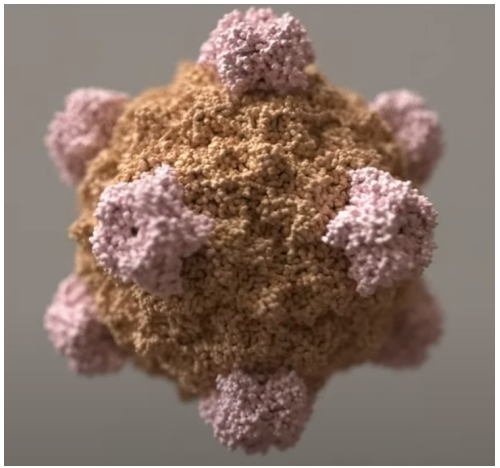
Objects: intuition

- Depending on application, we would like our 3D scene to contain various objects
- How do we even start with this?

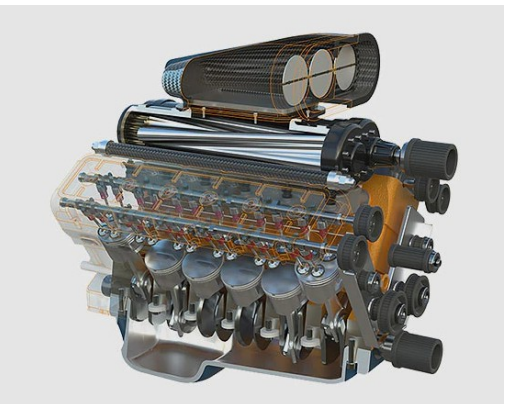
Arhitecture, Revit: https://www.autodesk.com/products/revit



Production examples at dreamworks, OpenVDB: https://www.openvdb.org/



Brady Johnston, Visaulising viruses: https://www.youtube.com/watch?v=adhTmwYwOiA&ab_channel=Blender



Autodesk Fusion 360: https://www.autodesk.com/products/fusion-360/overview

Objects: shape and material

For 3D modeling and rendering, real-world phenomena is decoupled into:

- **Shape: geometry**
  - **Modeling**: process of creating 3D objects by defining size, form, relative position and proportion to other objects
  - **Rendering**: solving visibility problem
- **Appearance: material**
  - **Modeling**: process of achieving desired object appearance
  - **Rendering**: calculating light-surface interaction → shading

https://polyhaven.com/models

Objects: shape representation

Object shape representations

- **Points**
  - Point clouds
  - Particle systems
- **Lines/curves**
  - Parametric curves
- **Surfaces**
  - Polygonal mesh
  - Subdivision surfaces
  - Parametric surfaces
  - Implicit surfaces
- **Volumetric objects/solids**
  - Voxels
  - Space partitioning data-structures

Modeling objects in 3D scene requires different shape representations.

- RenderMan
  - Geometry
    - Curves
    - NURBS
    - Polygons
    - Subdivision Surfaces
    - Particles
    - Volumes
    - Implicit Surfaces
    - Instancing
    - Aggregate Volumes
    - Modeling Guidelines

https://rmanwiki.pixar.com/display/REN24/Geometry

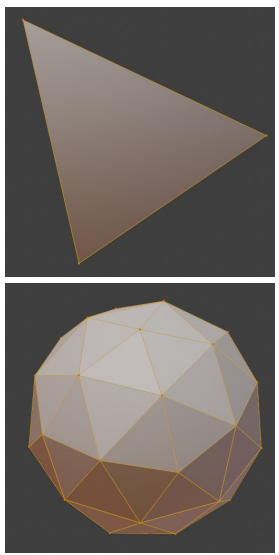
Object shape representations

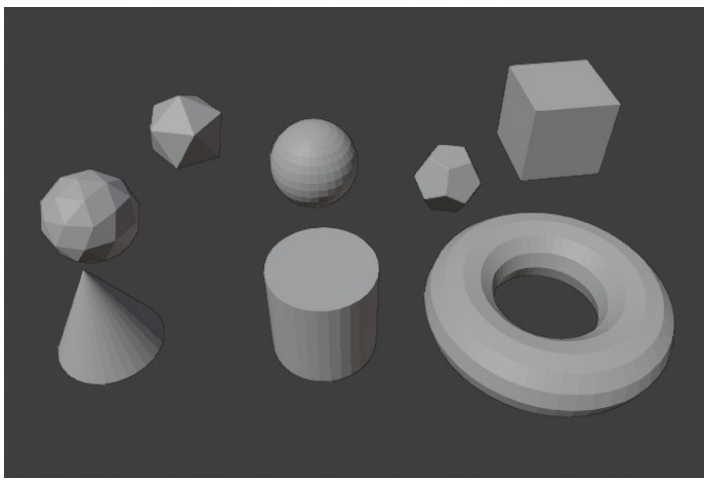
- Some representations are more convenient for **user and modeling**
  - Example: modeling curved surface (e.g., car) is very hard by using triangles
  - In 3D modeling tools, user is provided with various shape representations to work with
- Some representations are more suitable for **rendering**
  - Having one shape representation for rendering enables more efficient rendering procedures
  - Prior rendering, different representations are transformed into triangles*

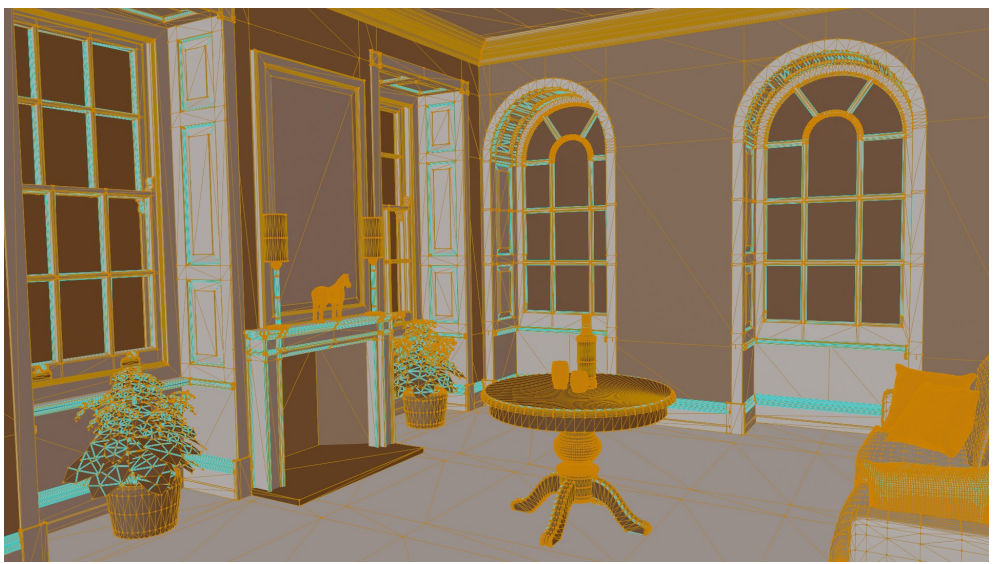
* Not all real-world phenomena can be well represented with triangles. Especially volumetric effects such as smoke or clouds. For such cases ray-casting and ray-tracing are often used.

Shape representation 1: triangles

- Simplest way to define surface is using **triangles**
  - Polygons with three points all lying on a plane (co-planar)
  - Widely used surface **shape representation primitive -** especially in rasterization-based rendering engines, e.g. Unity (https://unity.com/)
  - **Approximation primitive**: very well approximating various objects, e.g., curved surfaces
  - Very **simple**, great properties for **easy calculation**, often used for **efficient rendering** purposes
  - Not intuitive for modeling, thus different shape representations are introduced

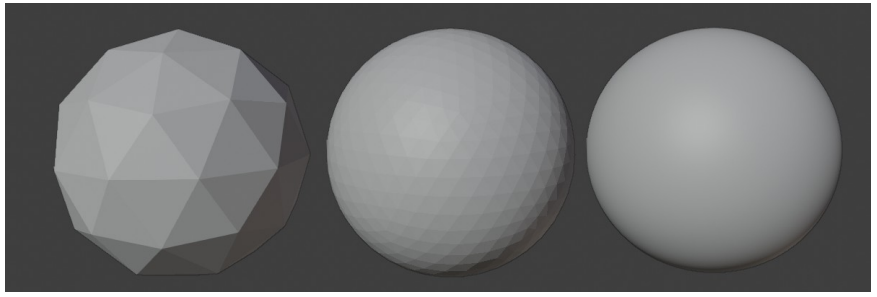




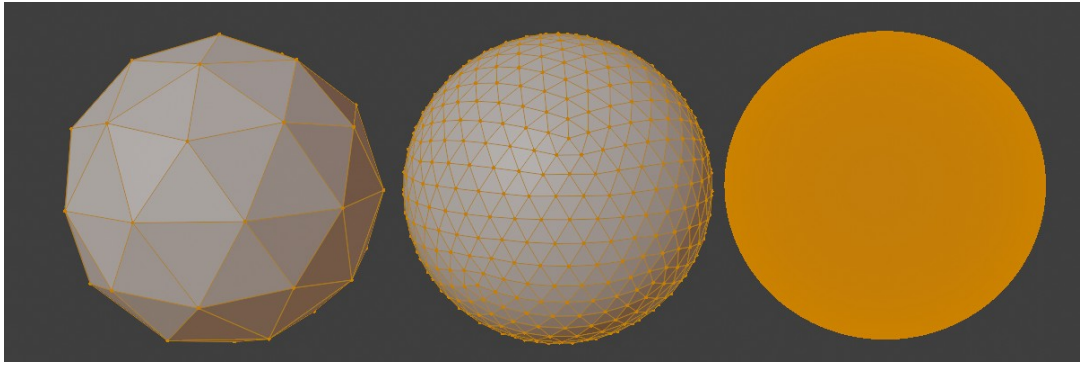


Shape representation 1: triangles

- Different shape representations can be converted to triangles:
  - **Triangulation:** converting shape to triangles
  - **Tessellation:** converting shape to polygons (of any sort)
- Any object can be represented using triangles
  - Higher quality approximation of curved surfaces requires more smaller triangles better fitting surface
  - Amount of details increase realistic representation but also complexity of rendering. Computer graphics often deals with trade-off between visual quality and speed*.



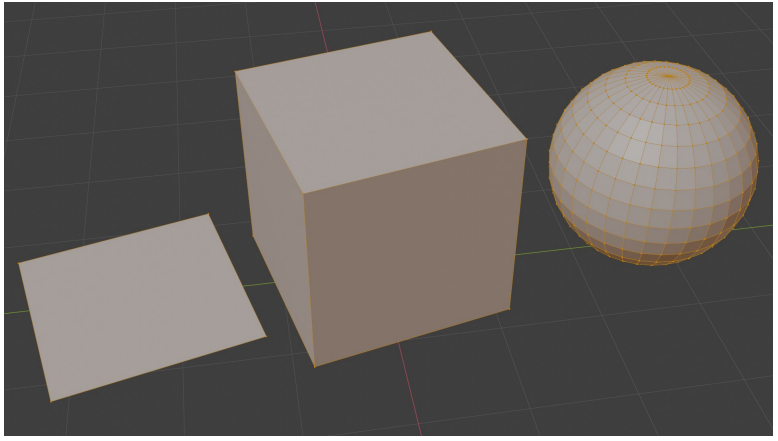
https://www.youtube.com/watch?v=x4RZRs3PD4Q&ab_channel=Udacity



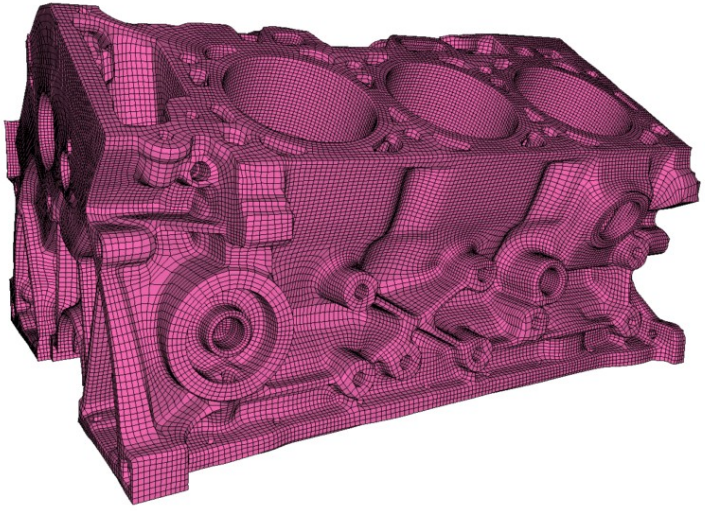
* Finding good tradeoff between visual quality and object complexity is big research field in computer graphics. Note also that distance of viewing plays important role in amount of detail that it has to be represented.

Shape representation 2: quad mesh

- Polygons with four vertices, not necessary co-planar
- Often provided by professional 3D modeling tools
- Often used for modeling (especially hard-surface modeling)



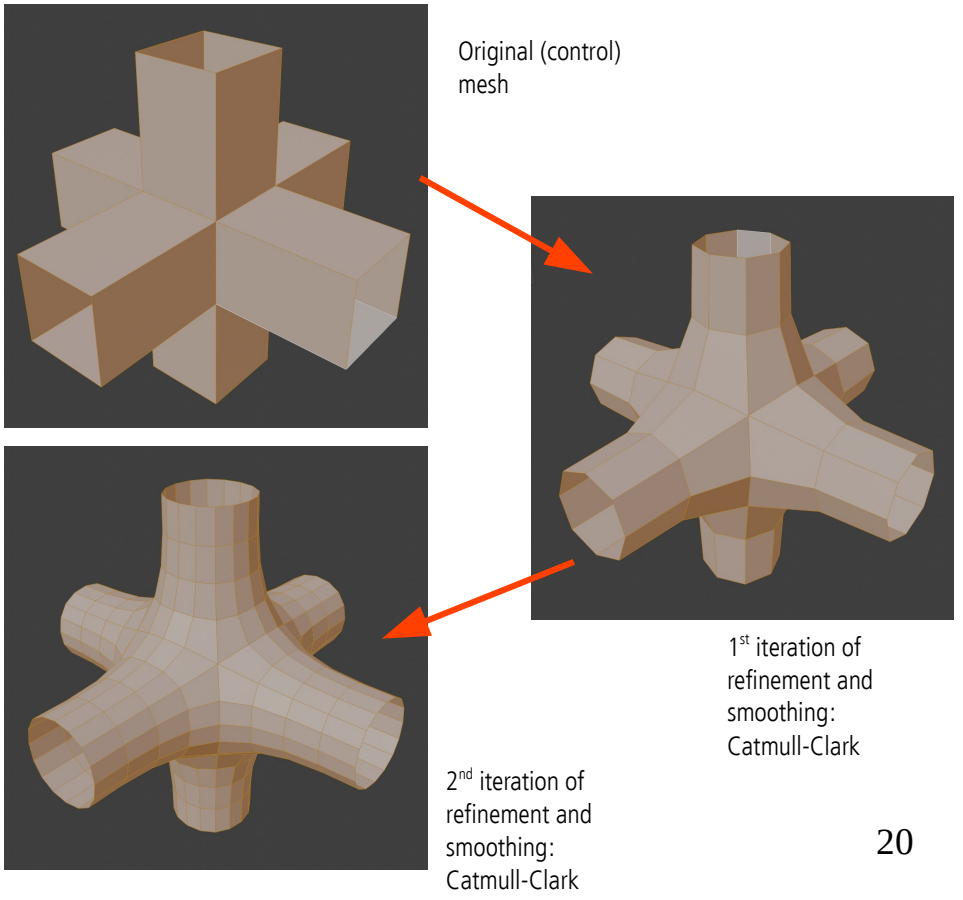
Blender mesh:
https://docs.blender.org/manual/en/latest/modeling/meshes/index.html



https://geometryfactory.com/products/igm-quad-meshing/

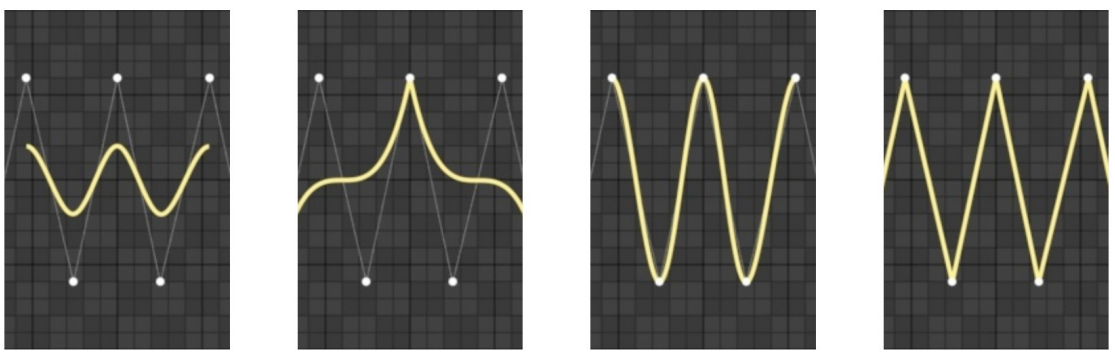
Meshes: subdivision surfaces

- Paradigm for defining **smooth and continuous surfaces** from polygon meshes
- Provides **infinite level of detail**: as many triangles as needed can be generated.
- Approaches:
  - Loop's method
  - Catmull-Clark subdivision



Shape representation 3: Parametric curves

- Useful for modeling large number of thin strands
  - Fur, hair, etc.
- Also used for animation
- Arbitrary, smooth curve shape is determined by control points and parametric equation.
- Methods: Linear, Bezier curve, B-Spline, Catmull-Rom



B-Spline | Bezier | Catmull-Rom | Linear

https://rmanwiki.pixar.com/display/REN24/Curves

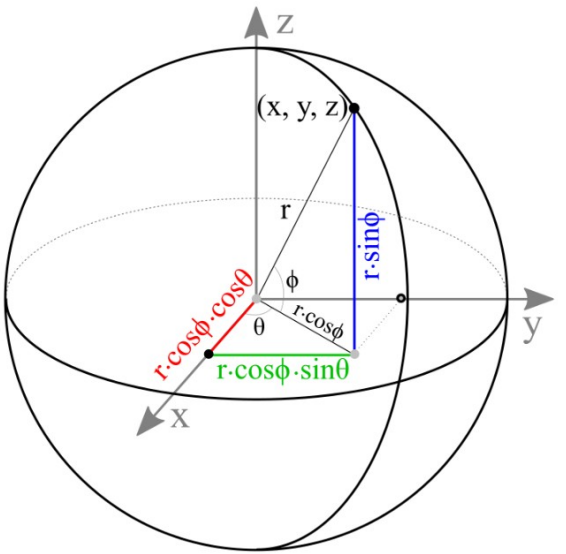
Shape representation 4: parametric surfaces

- **Simple shapes can be described mathematically** using parametric equations, e.g., spheres, disks, planes
- Equation of sphere with origin $C(c_x, c_y, c_z)$ and radius r:

$$(x-c_x)^2 + (y-c_y)^2 + (z-c_z)^2 = r^2$$

- Arbitrary point on sphere is computed using parametric equations:

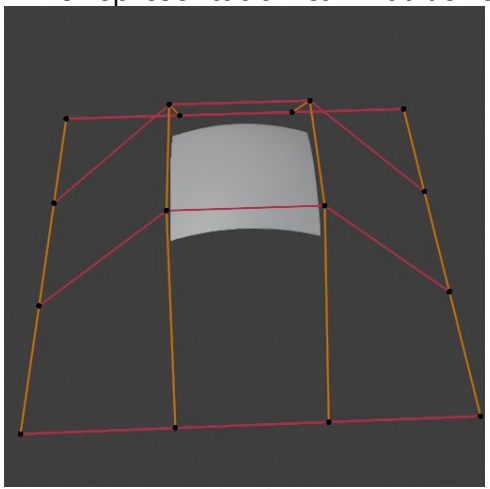
$$\begin{aligned} x &= (r \cdot \cos\phi) \cdot \cos\theta \\ y &= (r \cdot \cos\phi) \cdot \sin\theta \\ z &= r \cdot \sin\phi \end{aligned}$$



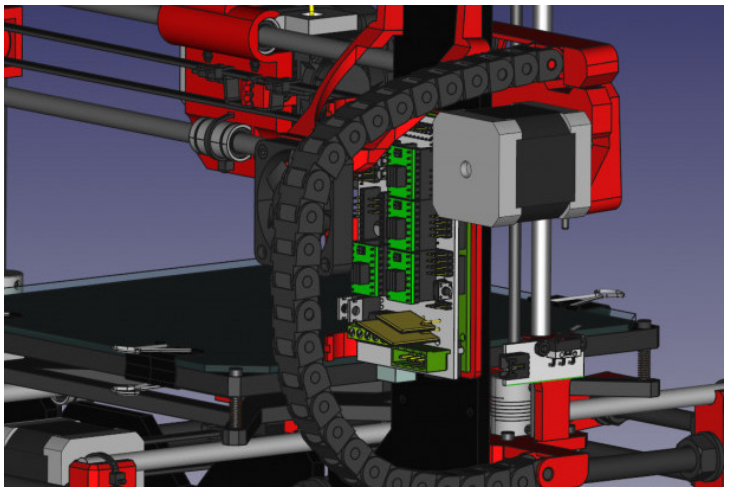
More on spheres in OpenGL:
http://www.songho.ca/opengl/gl_sphere.html

Shape representation 4: parametric surfaces

- More complex surfaces are represented by **points** which define a **control mesh** from which perfect curved surface can be generated using **parametric description**.
  - Note that this kind of representation is much more compact than polygonal mesh.
- Approaches: Bezier patches, B-Spline surfaces, NURBS, etc.
  - Often used in CAD software, e.g., design of objects for manufacturing
- This representation can not be rendered directly. **Tessellation** must be performed prior rendering.



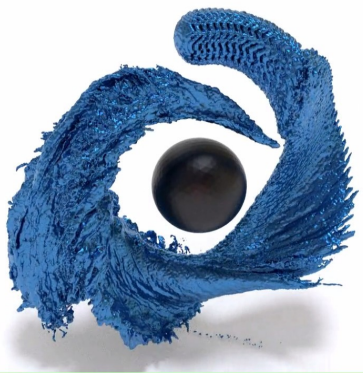
NURBS surface Blender:
https://docs.blender.org/manual/en/latest/modeling/surfaces/index.html



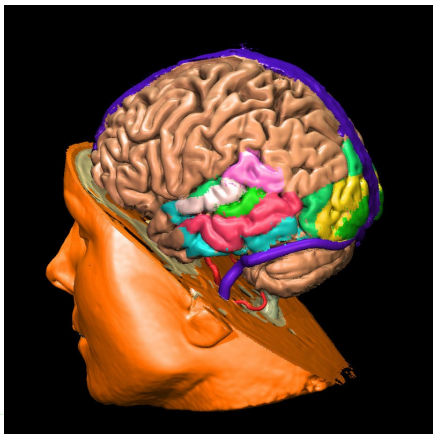
FreeCAD: https://www.freecadweb.org/features.php

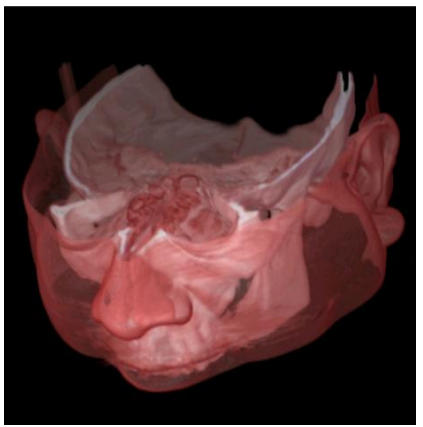
Shape representation 5: voxels

- **Voxel**: represents volume of space, e.g., cube in 3D uniform regular grid (index of the grid determines voxel location)
- Used for representing **volumetric data** (not only surface as in e.g., polygonal mesh)
  - Example: fluid (smoke or liquid) simulation requires computation on grid of cells – voxels
- Each voxel can contain various information:
  - Density or opacity information, e.g., medical applications, CT acquisition
  - One bit representing if its center is inside or outside surface, e.g., scanning or modeling

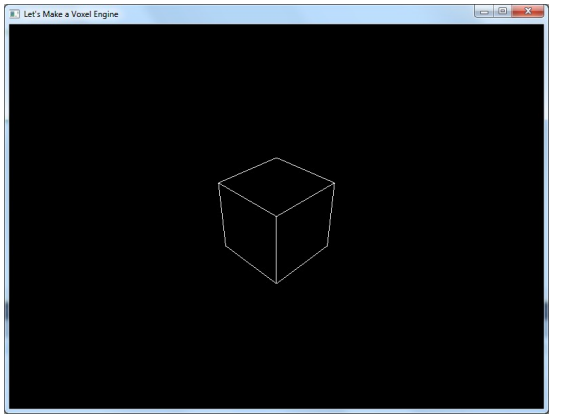


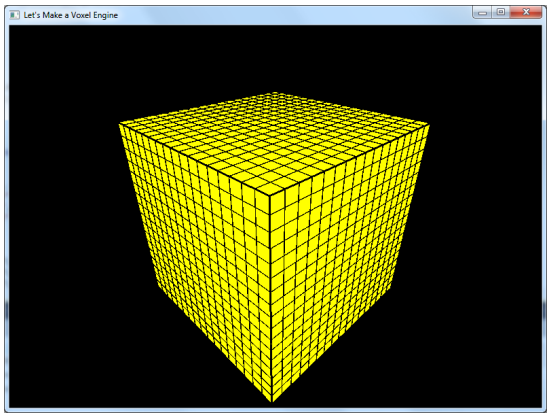
https://github.com/rlguy/GridFluidSim3D





Voxels in medicine: https://www.voxel-man.com/3d-navigators/downloads/
http://ablesw.com/3d-doctor/volume.html





Getting started with voxels:
https://sites.google.com/site/letsmakeavoxelengine/home/water

Shape representation 5: voxels

Voxel-based modeling: https://ephtracy.github.io/

Smoke simulation in Blender:
https://docs.blender.org/manual/en/3.4/physics/fluid/index.html
https://www.youtube.com/watch?v=zyIJQHlFQs0&ab_channel=Polyfjord

- **Rendering**: ray-tracing or converting to polygons using maching cubes algorithm.
- Large voxel grids are **memory expensive**.
  - Often, not all voxels in grid are needed and thus memory lighter sparse-voxel representations, e.g., sparse voxel octrees are used: https://www.nvidia.com/docs/IO/88972/nvr-2010-001.pdf

Shape representation 5: voxels

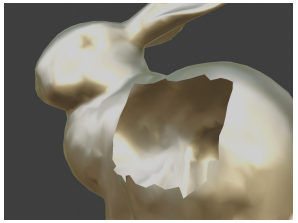
Voxel art: https://www.artstation.com/madmaraca

Shape representation 5: voxels

https://research.dreamworks.com/research_library/cloud-modeling-and-rendering-for-puss-in-boots/

Shape representation: surface and volume

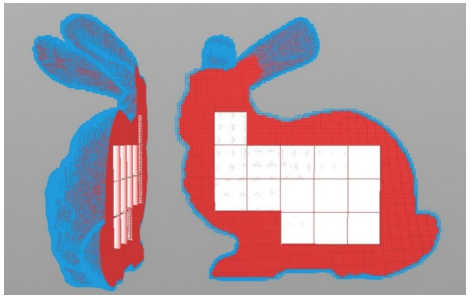
- Shape representation can be used to describe:
  - **Surfaces**, e.g., mesh or parametric surfaces
  - **Volumes**, e.g., voxels

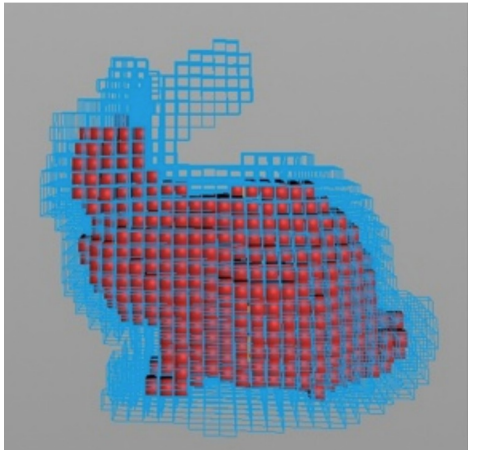


Example: for opaque objects, we don't care about object interior (approximated with material model)

Example: for smoke or other participating media, information inside object is important (e.g., density of smoke)

Volume rendering:
https://help.maxon.net/c4d/s26/en-us/Content/_REDSHIFT_/html/Volume+Rendering.html





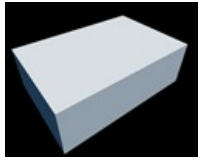
Production examples at dreamworks, OpenVDB:
https://artifacts.aswf.io/io/aswf/openvdb/openvdb_production_2015/1.0.0/openvdb_production_2015-1.0.0.pdf

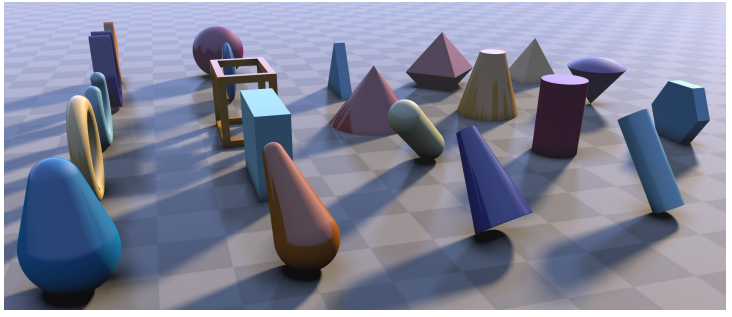
Shape representation 6: implicit

```
float sdSphere( vec3 p, float s )
{
  return length(p)-s;
}
```

- Point on surface is described with **implicit – signed distance – function (SDF):**
  `f(x,y,z) = f(p) = 0`
  - Point is on surface if `f(p) = 0`. If `f(p) < 0` then `p` inside, otherwise `p` is outside
- **Simple shapes** can be defined using SDFs: cubes, spheres, etc.

```
float sdBox( vec3 p, vec3 b )
{
  vec3 q = abs(p) - b;
  return length(max(q,0.0)) + min(max(q.x,max(q.y,q.z)),0.0);
}
```

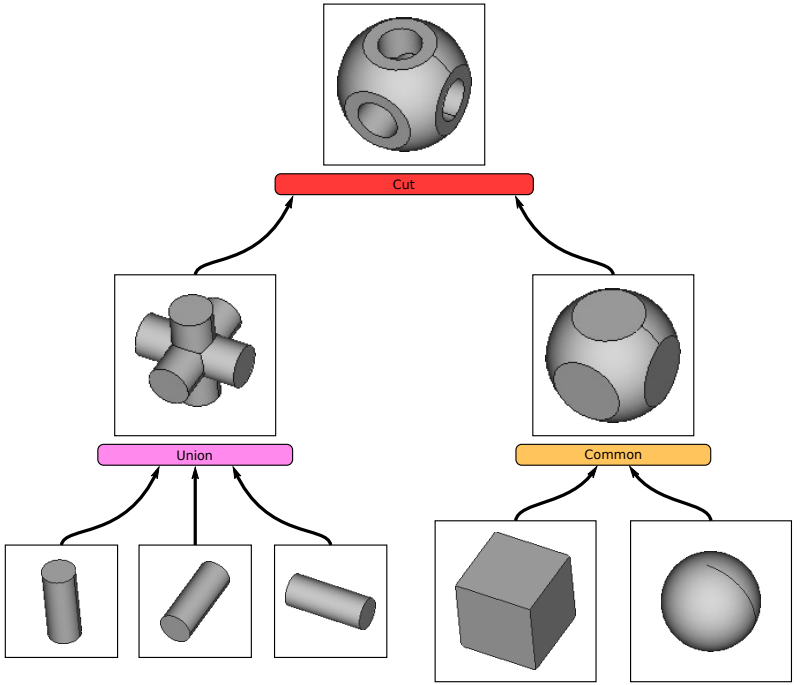




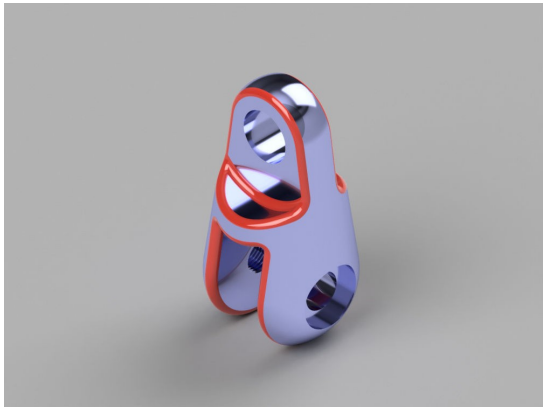
Implicit shape representations:
https://iquilezles.org/articles/distfunctions/
https://www.youtube.com/watch?v=62-pRVZuS5c&ab_channel=InigoQuilez

Shape representation 6: implicit

- **Modeling** with SDFs is done using **constructive solid geometry** operations: addition, subtraction, etc.
- **Rendering**
  - Converted to triangulated mesh using marching cubes algorithm
  - Ray-tracing (simple ray-implicit shape tests)
- Often used in engineering and CAD modeling environments

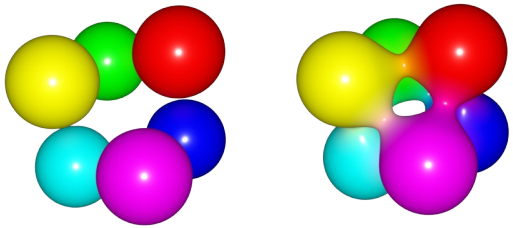


https://wiki.freecadweb.org/Constructive_solid_geometry



https://gallery.autodesk.com/projects/22791/constructive-solid-geometry-csg

Shape representation 6: implicit

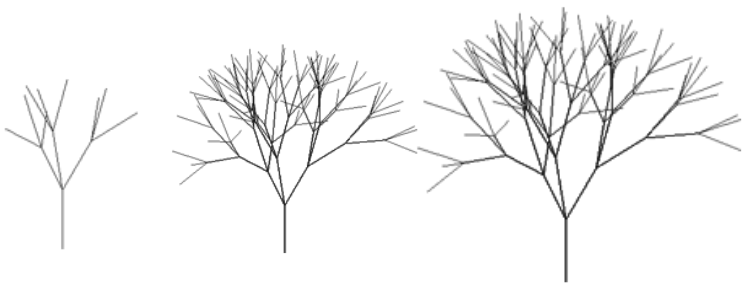


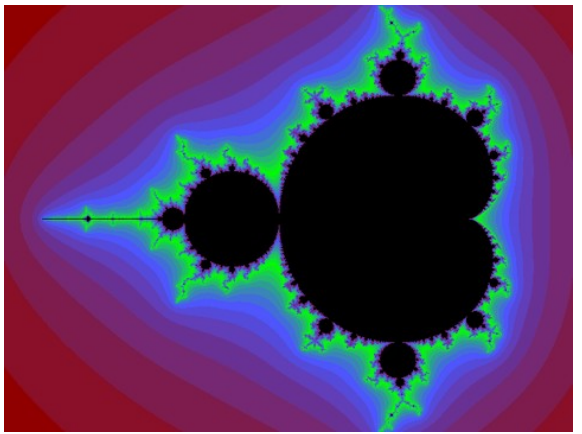
Good for representing organic objects (metaballs):
https://rmanwiki.pixar.com/display/REN24/Implicit+Surfaces
https://docs.blender.org/manual/en/latest/modeling/metas/introduction.html

Implicit representation for modeling and rendering:
https://www.gdcvault.com/play/1025316/Advanced-Graphics-Techniques-Tutorial-GPU

Shape representation 7: fractals

- Irregular geometrical shapes with symmetrical property
  - Complex, natural phenomena such as clouds, mountains, trees, vegetation, galaxies can be represented using fractals
- Generative art: http://blog.hvidtfeldts.net/

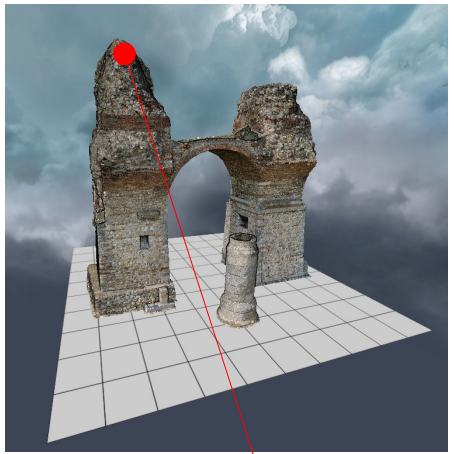




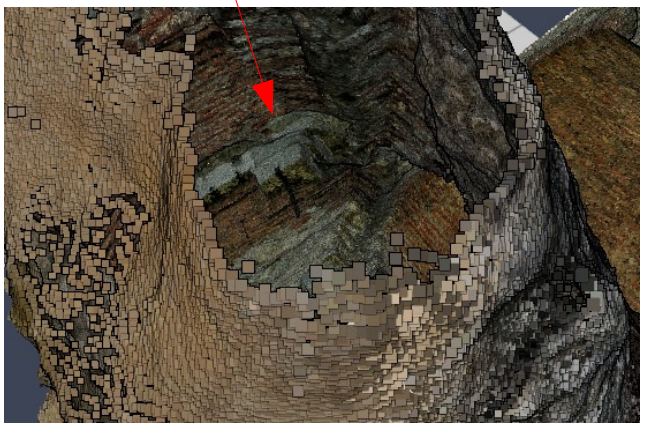
L-Systems: grammar for complex shapes generation using iterations:
https://www.sidefx.com/docs/houdini/nodes/sop/lsystem.html
https://www.youtube.com/watch?v=0vE8GiXhOWM&t=1248s&ab_channel=Houdini
https://www.jonathanrpace.com/project_trees.html

Mandelbrot: https://www.mandelbulb.com/
https://www.geeks3d.com/20091116/exploration-of-the-real-3d-mandelbrot-fractal/

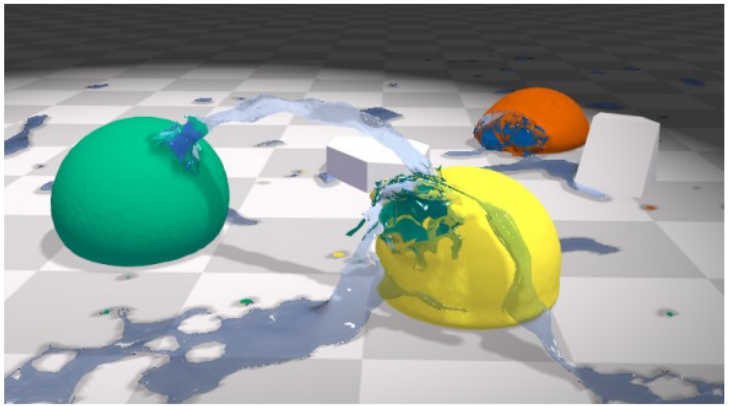
Shape representation 7: Points

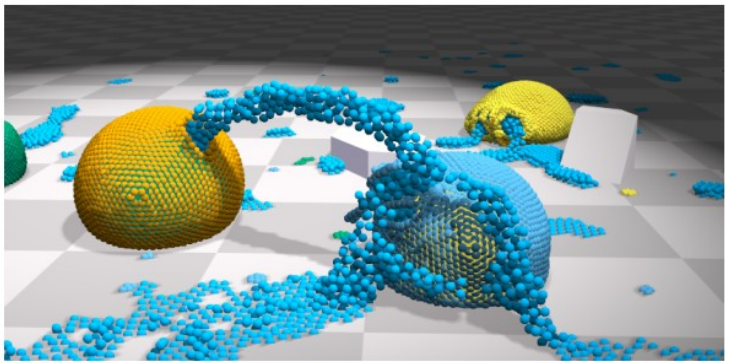


Scanned real-world objects can be stored as **point clouds**



https://github.com/potree/potree



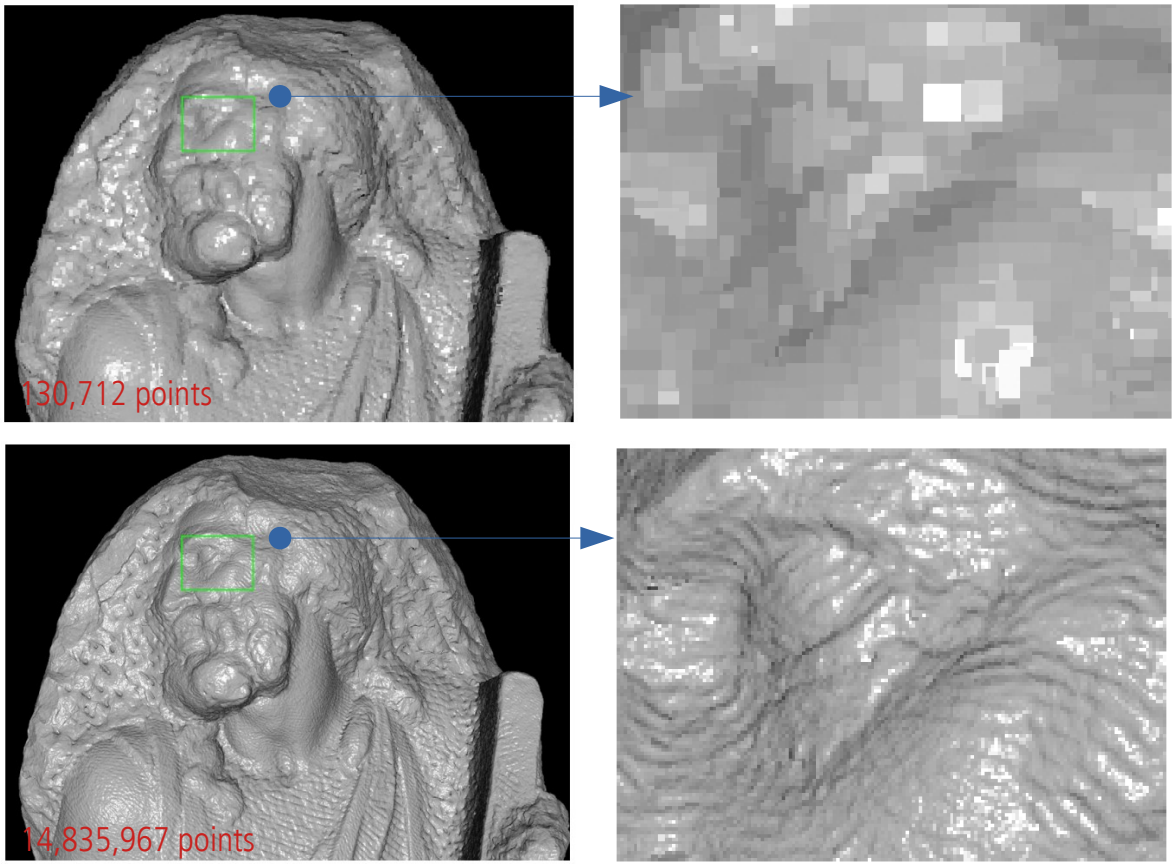


Unified physics for real-time applications using points:
https://mmacklin.com/uppfrta_preprint.pdf

Shape representation 7: Points

- Point clouds rendering:
  - **Splat** – shape with radius
  - **Surfels** – point based surface element*
  - Conversion to mesh

Points and implicit surfaces:
https://www.youtube.com/watch?v=u9KNtnCZDMI&ab_channel=MediaMolecule

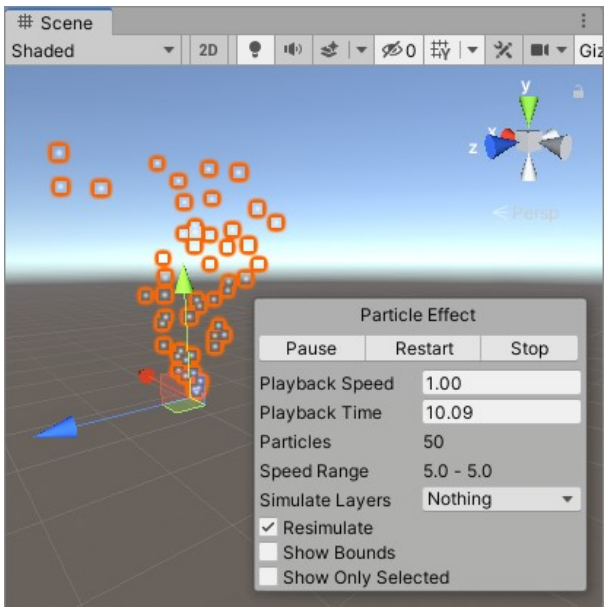


Qsplat: http://graphics.stanford.edu/software/qsplat/

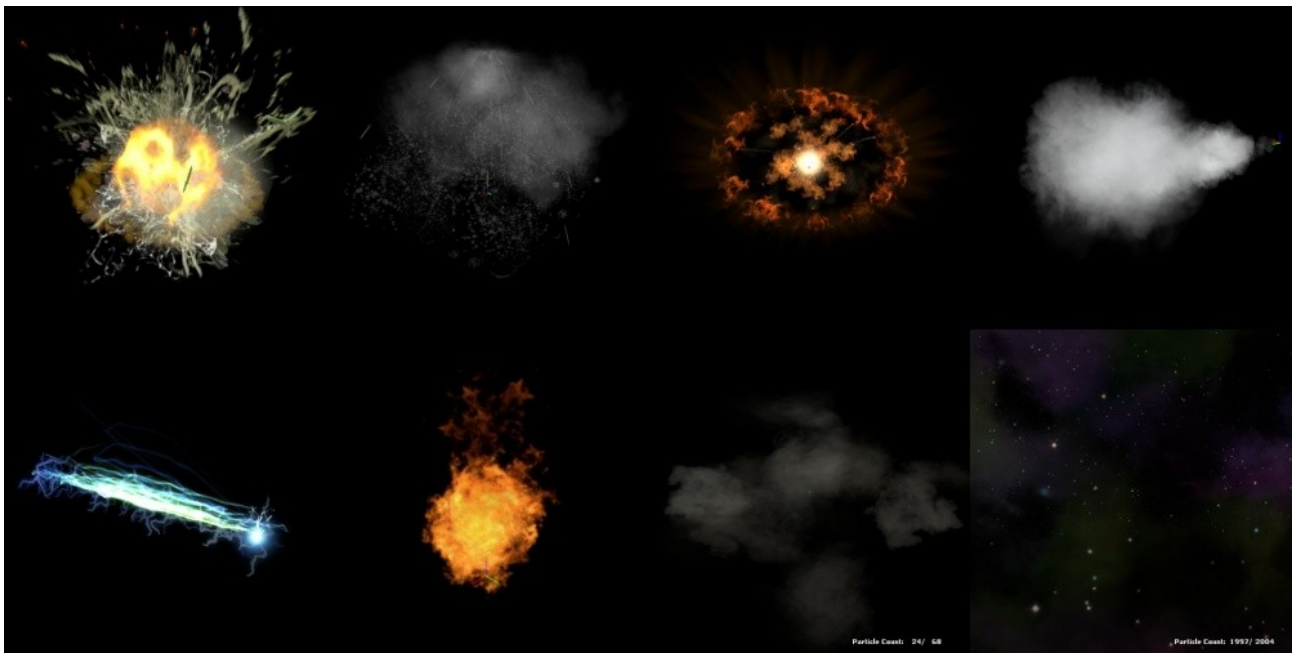
* Surfels:
https://vcg.seas.harvard.edu/publications/surfels-surface-elements-as-rendering-primitives

Shape representation 8: particle systems

- Collection of small objects that are set into motion using some algorithm.
  - Often, **billboards** – image that are always oriented towards camera is used as particle particles
- Applications: simulation of fire, smoke, water, FX, etc.



https://docs.unity3d.com/ScriptReference/ParticleSystem.html



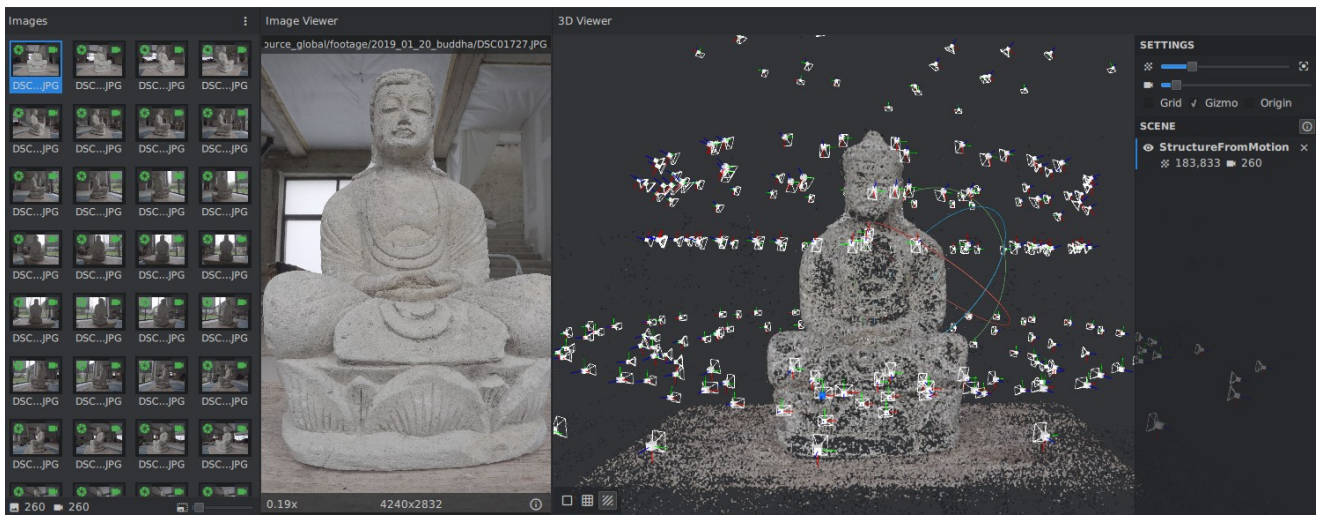
https://developer.valvesoftware.com/wiki/Particle_System_Overview

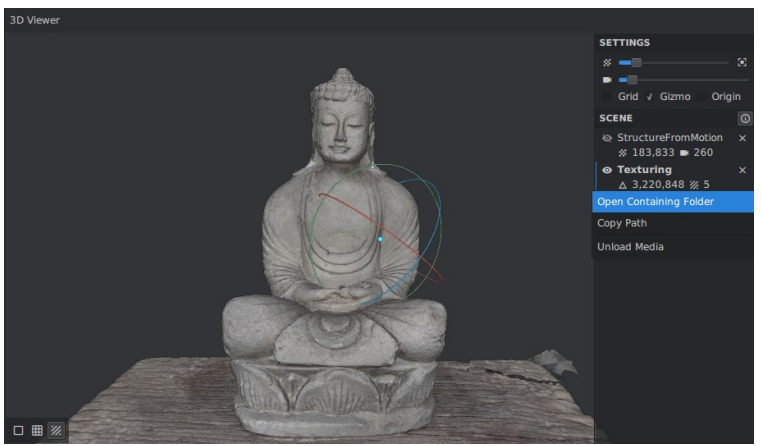
Various shape representations?

- User (modeling) point of view: shape representations must be intuitive and flexible to use and cover different applications
  - Acquisition, modeling, animation, texturing, analysis, etc.
- Rendering point of view: efficient visibility calculation

Applications

- **Acquisition**:
  - Photogrammetry (reconstructing from photographs)
    - Point clouds, meshes
  - Scanning: CT, MRI, Kinect



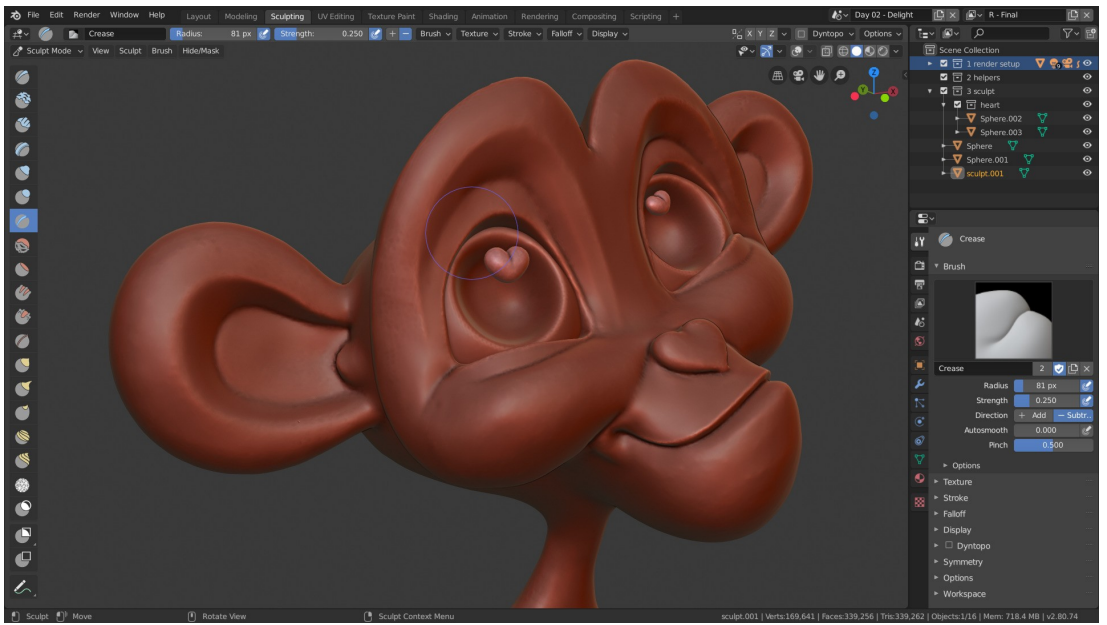


Photogrammetry: given object images, point cloud and mesh are constructed: https://sketchfab.com/blogs/community/tutorial-meshroom-for-beginners

Applications

- **Modeling: Sculpting**
  - Mesh, voxels, implicit representations

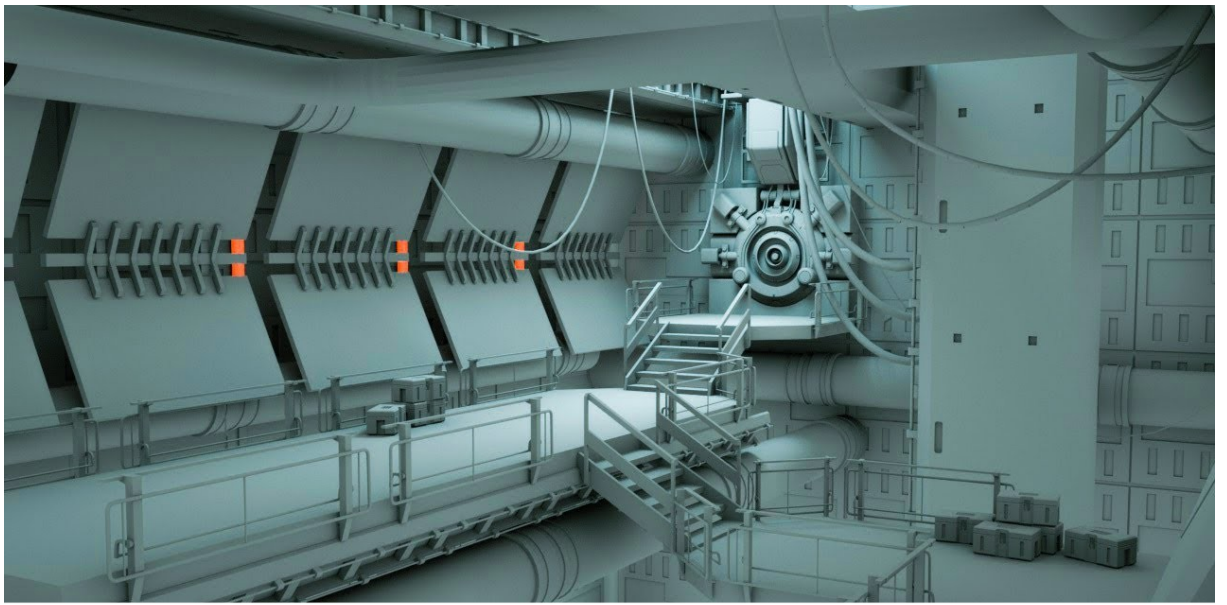
Jasmine, Zbrush,
http://pixologic.com/zbrush/gallery/?year=2018



Blender, https://www.blender.org/features/sculpting/

Applications

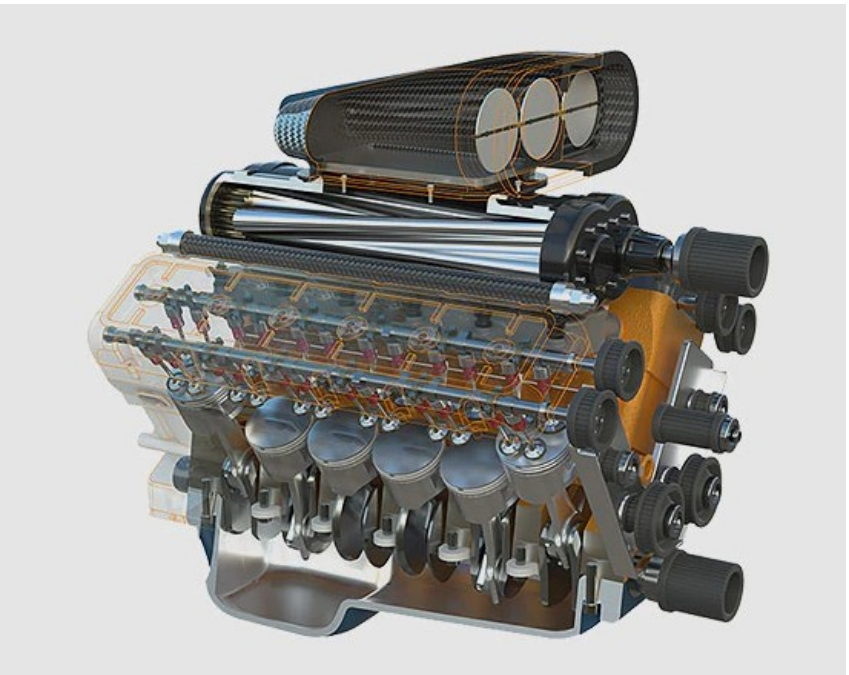
- **Modeling: hard-surface modeling**
  - Quad mesh



Hard surface modeling:
https://www.sidefx.com/tutorials/houdini-modelling-tutorial-175-houdini-hard-surface-modelling-tutorial/
Hard surface modeling: https://www.youtube.com/watch?v=1qVbGr_ie30&ab_channel=JoshGambrell

Applications

- **Modeling: curved surfaces**
  - Parametric surfaces



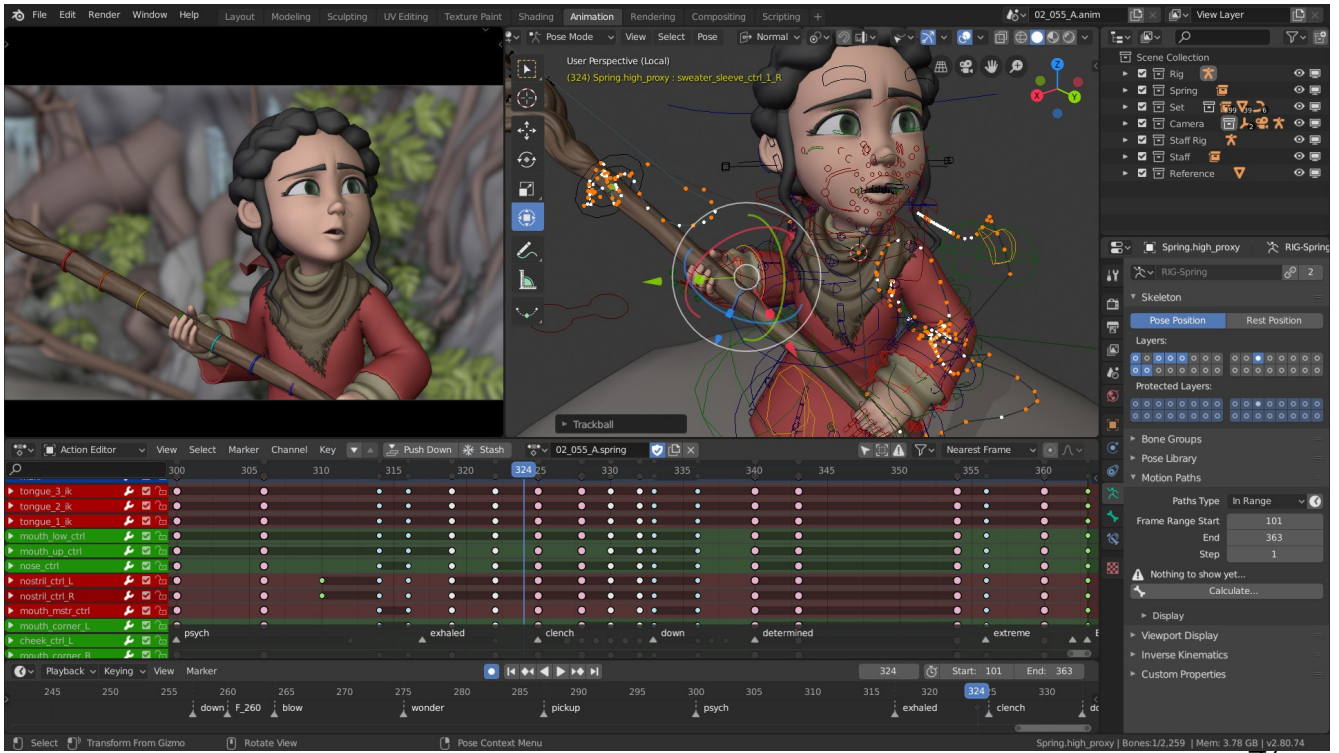
Renderman NURBS: https://rmanwiki.pixar.com/display/REN24/NURBS

Autodesk Fusion 360: https://www.autodesk.com/products/fusion-360/overview

Applications

Animation: manual

- Meshes, parametric surfaces

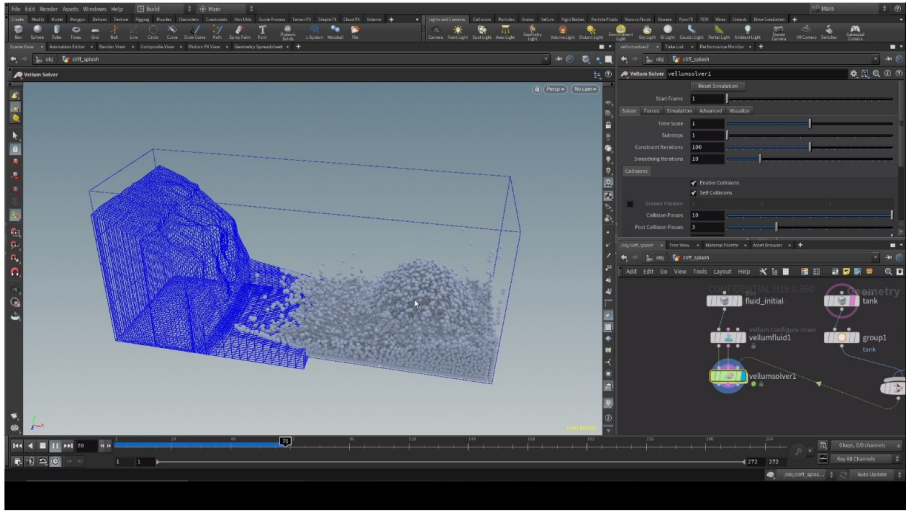


https://www.blender.org/features/animation/

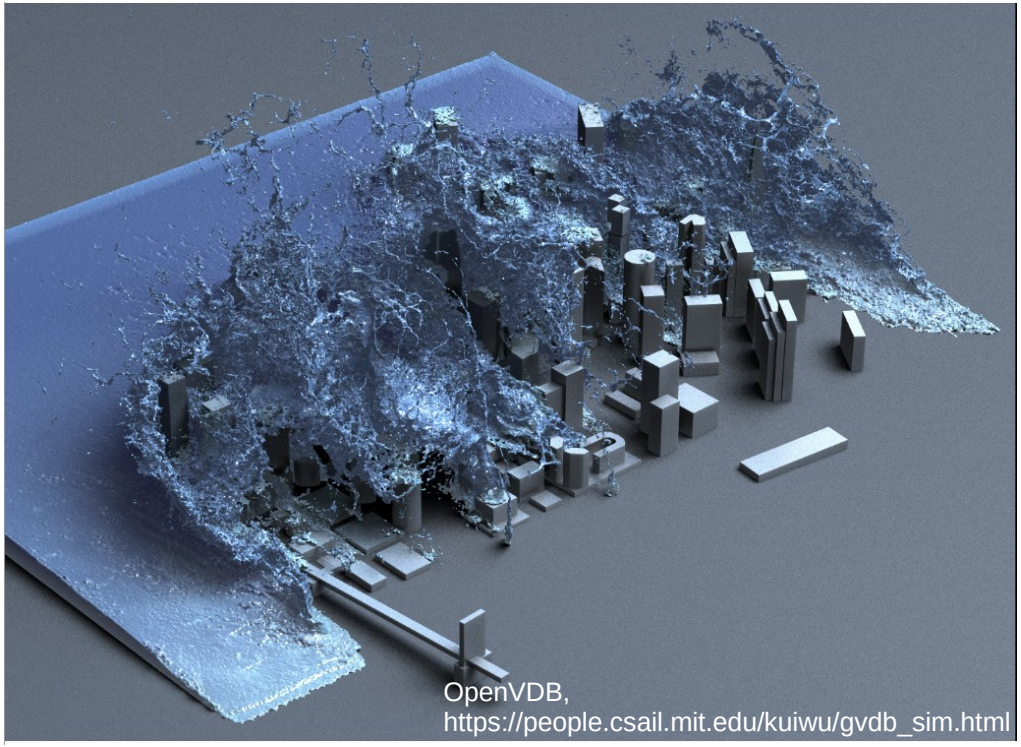
Applications

Animation: procedural, simulation

- Voxels, points



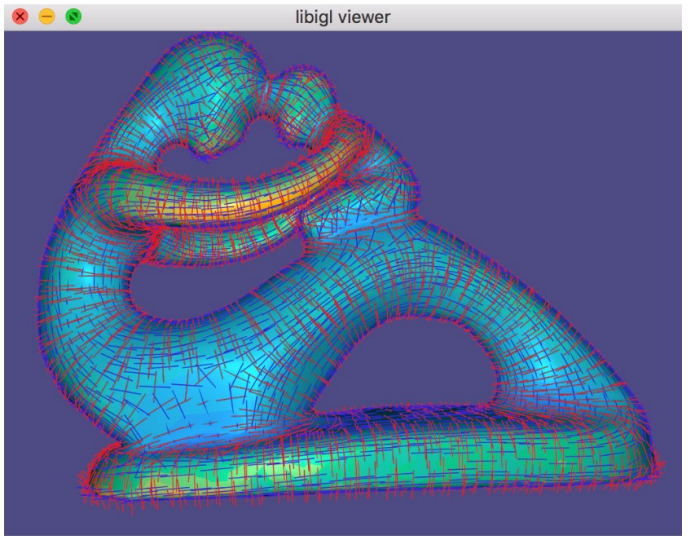
Houdini,
https://www.sidefx.com/tutorials/h19-vellum-fluids-starter-pack/



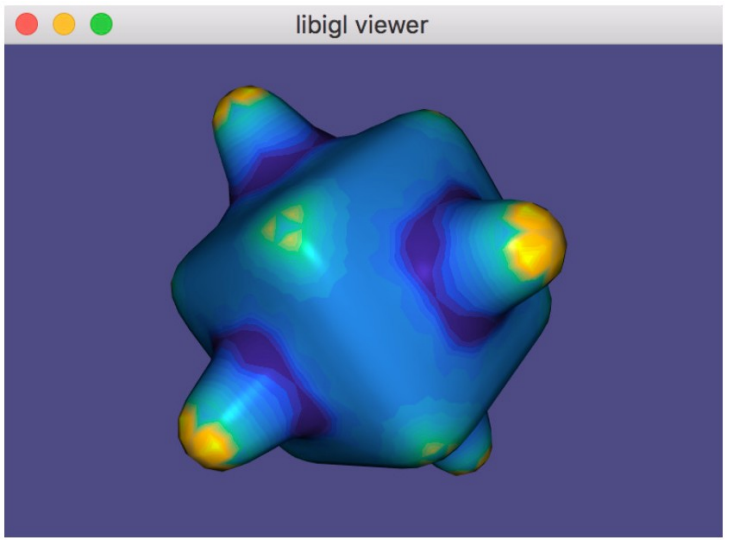
OpenVDB,
https://people.csail.mit.edu/kuiwu/gvdb_sim.html

Applications

- **Analysis and processing**
  - Meshes



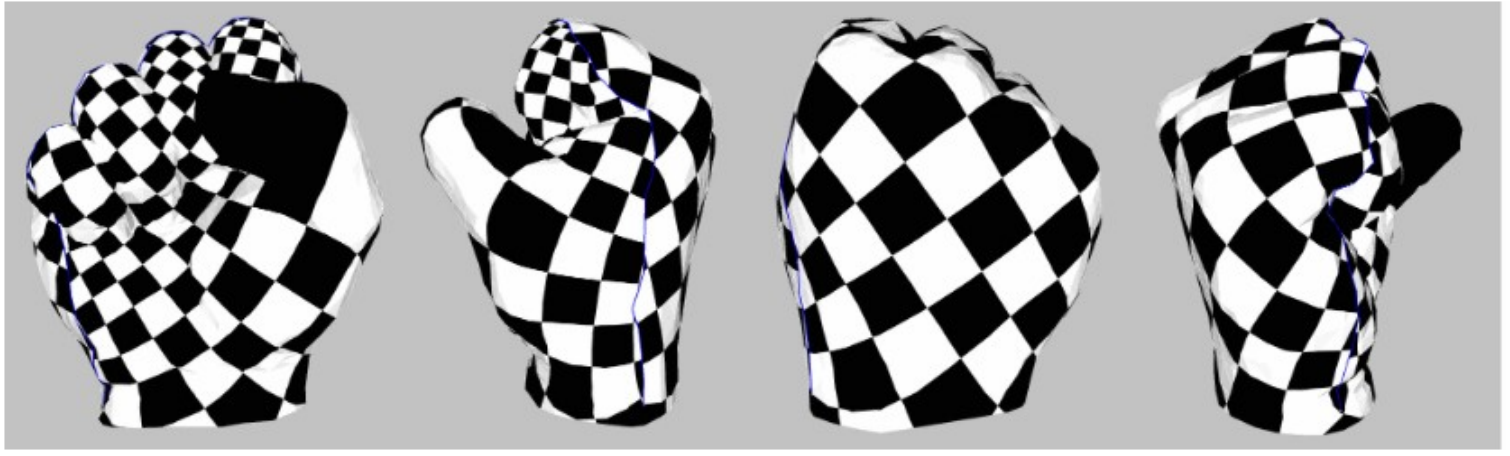
Example: principled curvature directions vectors (libigl: https://libigl.github.io/)

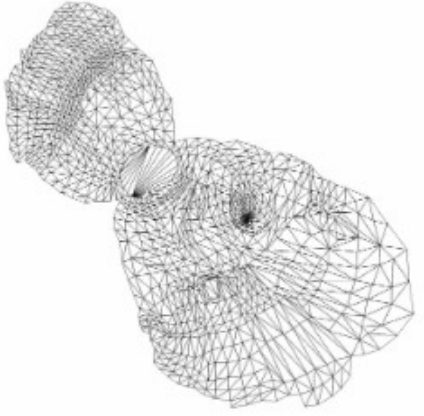


Example: Gaussian curvature visualization using pseudo color (libigl: https://libigl.github.io/)

Applications

- **Texturing**: parameterization
  - Mesh, parametric surfaces, implicit surfaces





Mesh representations require surface parametrization for texture application. CGAL: Least Squares Conformal Maps (https://www.cgal.org/)

Applications

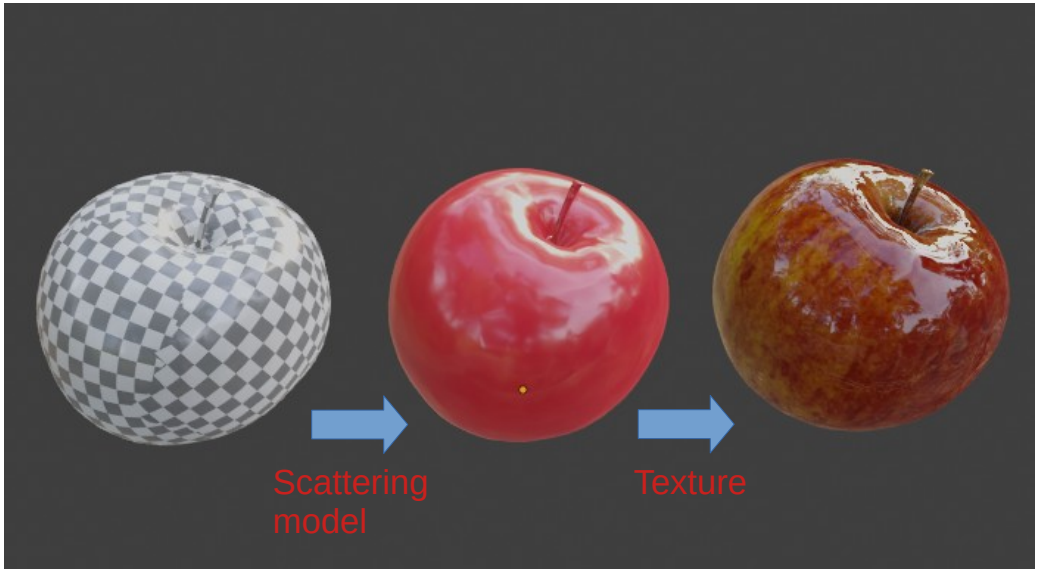
- **Rendering**: efficient visibility solving
  - Triangles, implicit surfaces

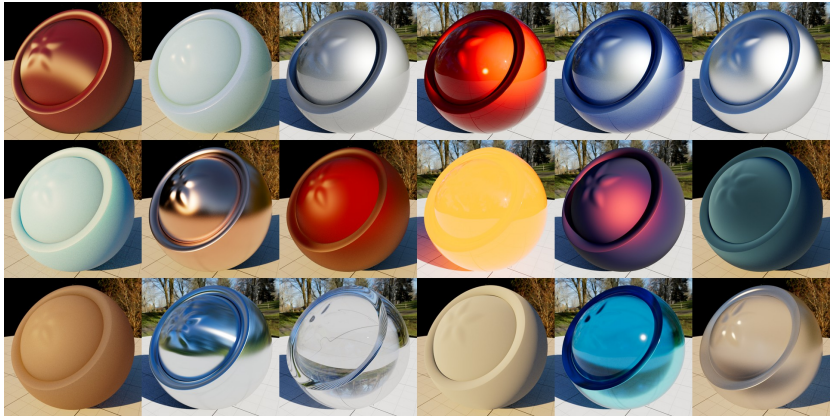
Objects: material

https://rmanwiki.pixar.com/display/REN24/Shading

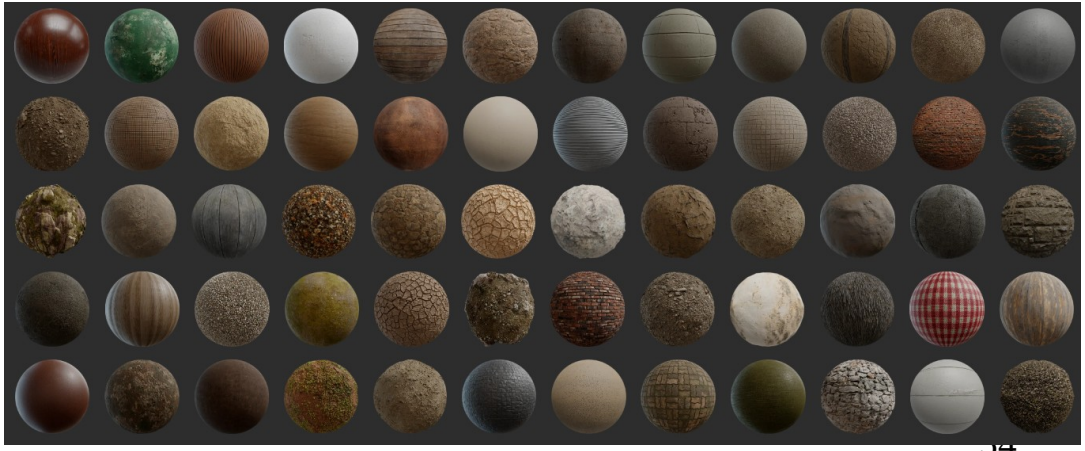
Objects: material

- **Modeling**: achieving desired appearance
  - Color, reflectivity
  - Surface patterns
- **Rendering**: crucial **shading** information; light-surface interaction
  - **Scattering function**
  - **Texture**: variation of scattering function properties



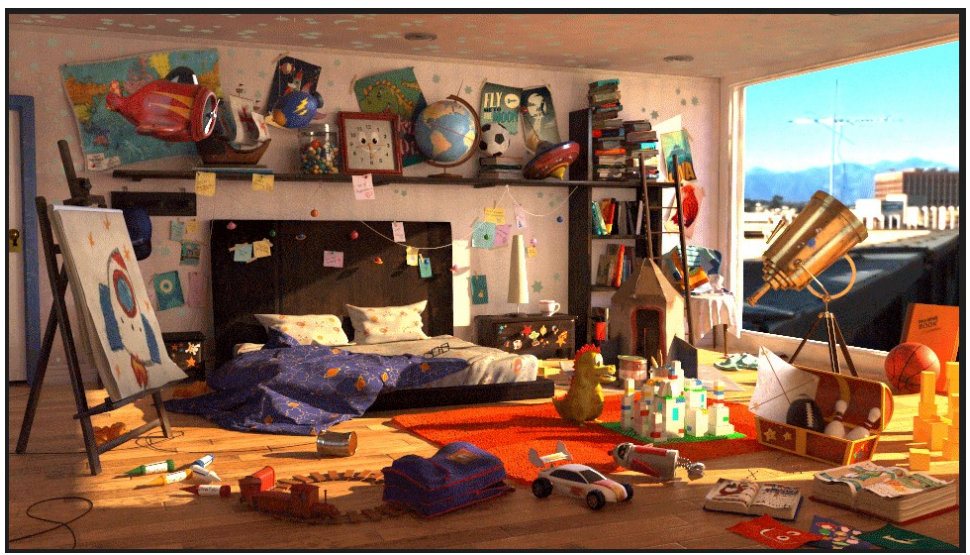


Scattering function: https://appleseedhq.net/gallery.html



Texture: https://polyhaven.com/textures

Lights

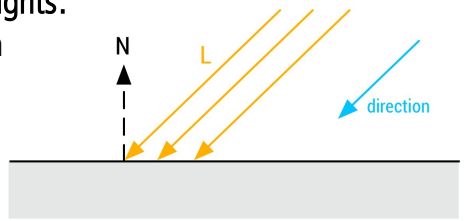


https://discover.therookies.co/2018/06/24/learn-how-to-light-and-render-like-a-pixar-artist/

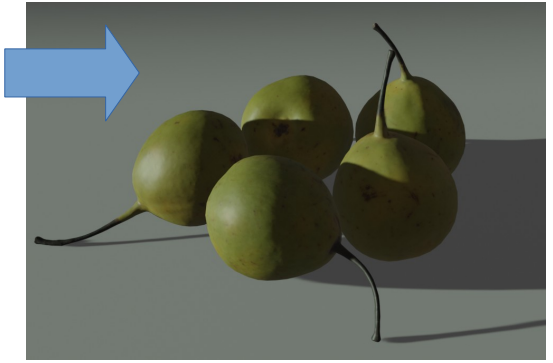
Lights: non-physical lights

Directional lights:

- Direction
- Intensity
- Color

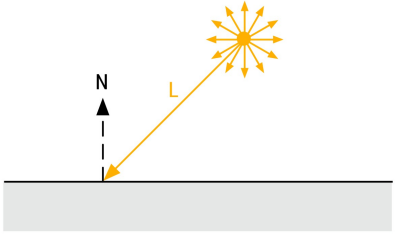


Real-world counterpart: sun



Point lights:

- Position
- Intensity
- Color



Real-world counterpart: light bulb

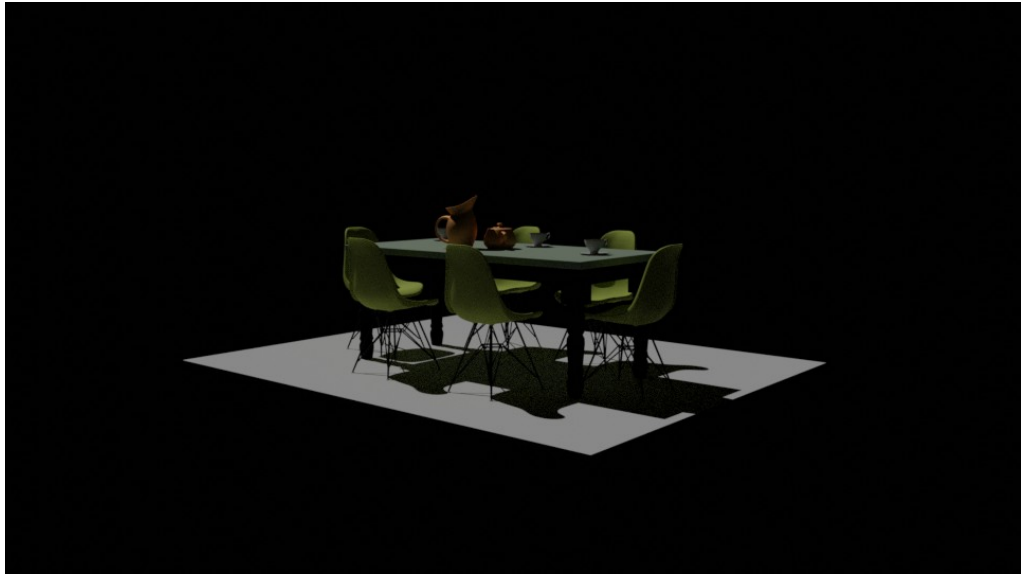
Lights: physical lights

- Next to position, intensity and color, physical light define:
  - **Shape** and **size**

Area light size determines softness of shadows (left: small area light, right: large area light) https://www.pbr-book.org/3ed-2018/Light_Sources/Area_Lights

Lights: environment light

- Besides local or directional lights, scene can be illumination from all directions using **environment illumination**.
- Real-world image can be acquired and wrapped around 3D scene → **environment mapping**.



https://polyhaven.com/hdris

Cameras

Camera model determines portion of visible scene and simulates real-world camera effects: depth of field, bokeh, motion blur, etc. https://rmanwiki.pixar.com/display/REN24/Cameras

Cameras

- Pinhole camera model
  - Focal length
  - Film size
- Portion and amount of visible scene
  - Viewing frustum
- Advanced:
  - Lens: depth of field
  - Shutter speed: motion blur

Literature

- https://github.com/lorentzo/IntroductionToComputerGraphics