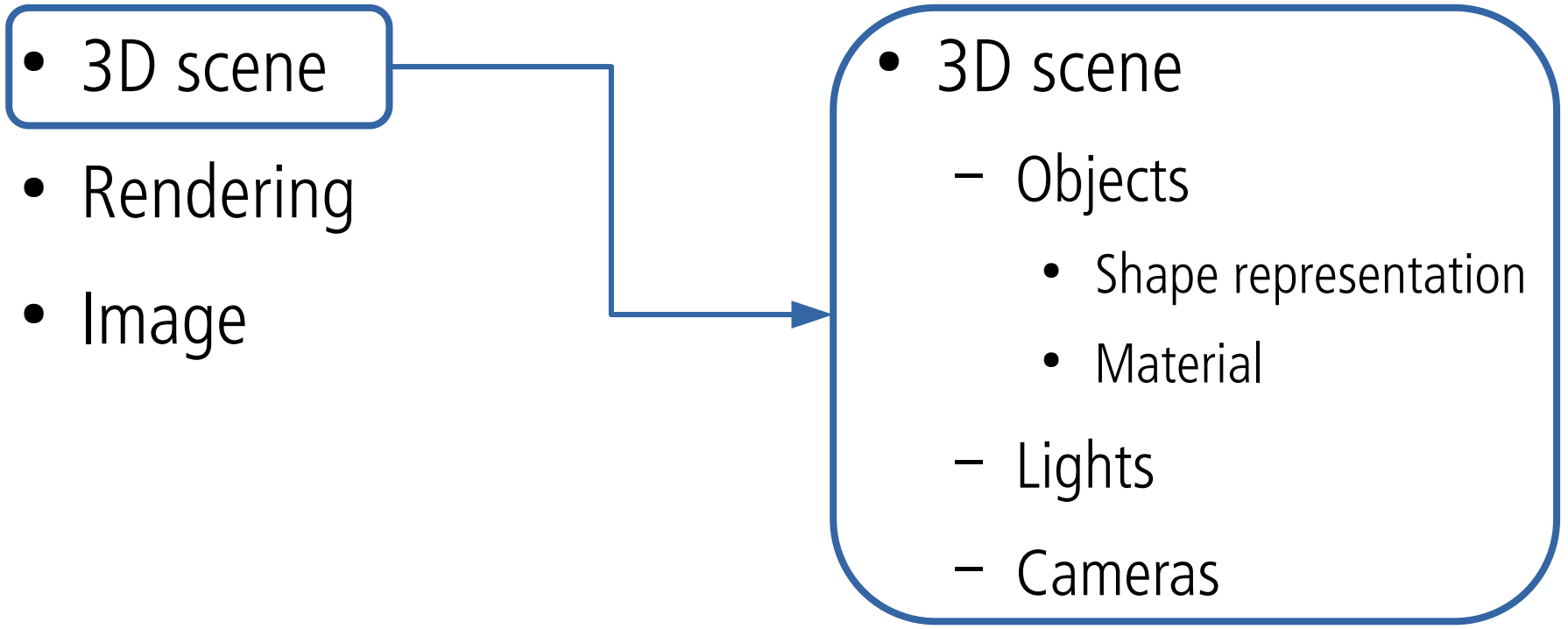
Lecture 1: 3D scene overview

DHBW, Computer Graphics

Lovro Bosnar

1.2.2023.

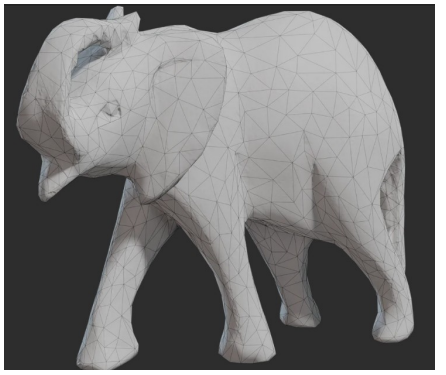
Syllabus


• 3D scene
• Rendering
• Image
• 3D scene
– Objects
• Shape representation
• Material
– Lights
– Cameras

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

A way of creating, acquiring, representing and manipulating:

- Objects
- Lights
- Cameras

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?

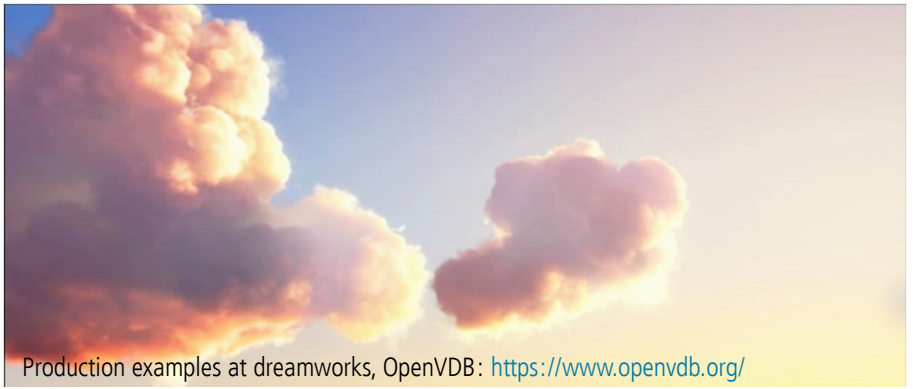
- How to use scene elements to create real-world phenomena and objects?
- How do we manipulate 3D models?

3D objects

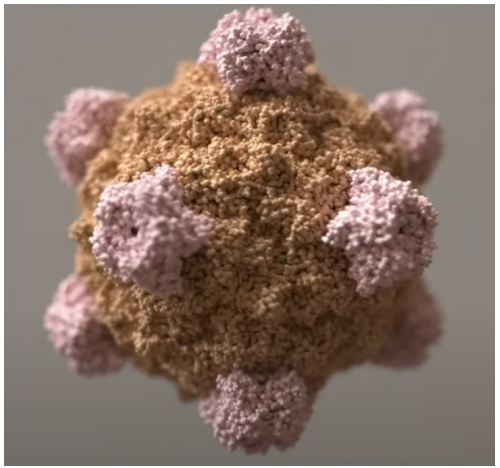
3D models: intuition

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

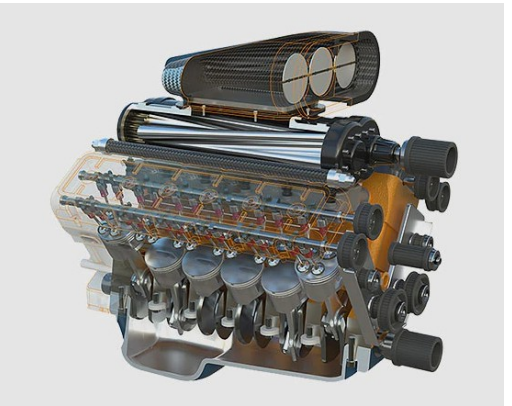
Arhitecture, Revit: https://www.autodesk.com/products/revit



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

3D models: shape and material

Decouple real-world phenomena, for 3D modeling, into:

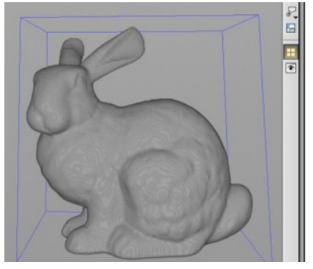
- **Shape: geometry**
  - Information: object size, form, relative position and proportion to other objects
  - Different representations: https://rmanwiki.pixar.com/display/REN24/Geometry
- **Appearance: material**
  - How objects appear when light falls on surface

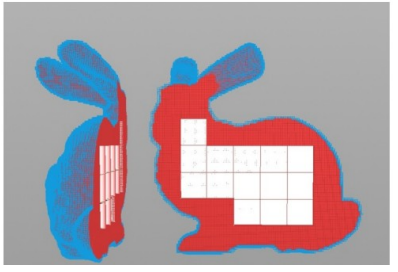
https://polyhaven.com/models

3D model shape representation

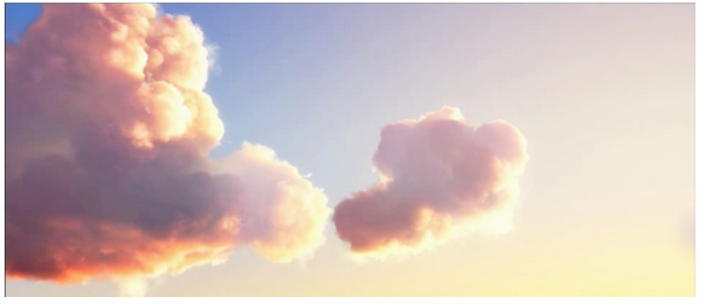
Shape representation: surface and volume

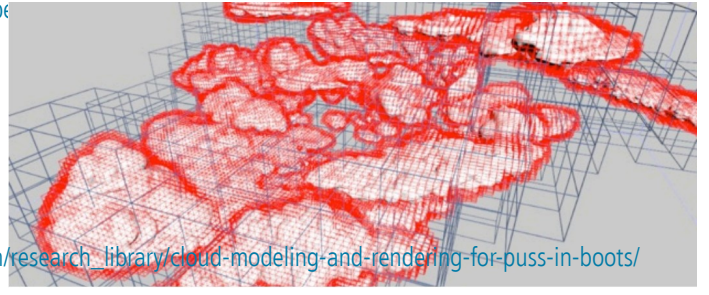
- Shape representation can be used to describe:
  - Surfaces
  - Volumes
- Decision on shape representations depends on phenomena that we are modeling





Production examples at dreamworks, OpenVDB:
https://artifacts.aswf.io/io/aswf/openvdb/openvdb_production_2015/1.0.0/ope
vdb_production_2015-1.0.0.pdf

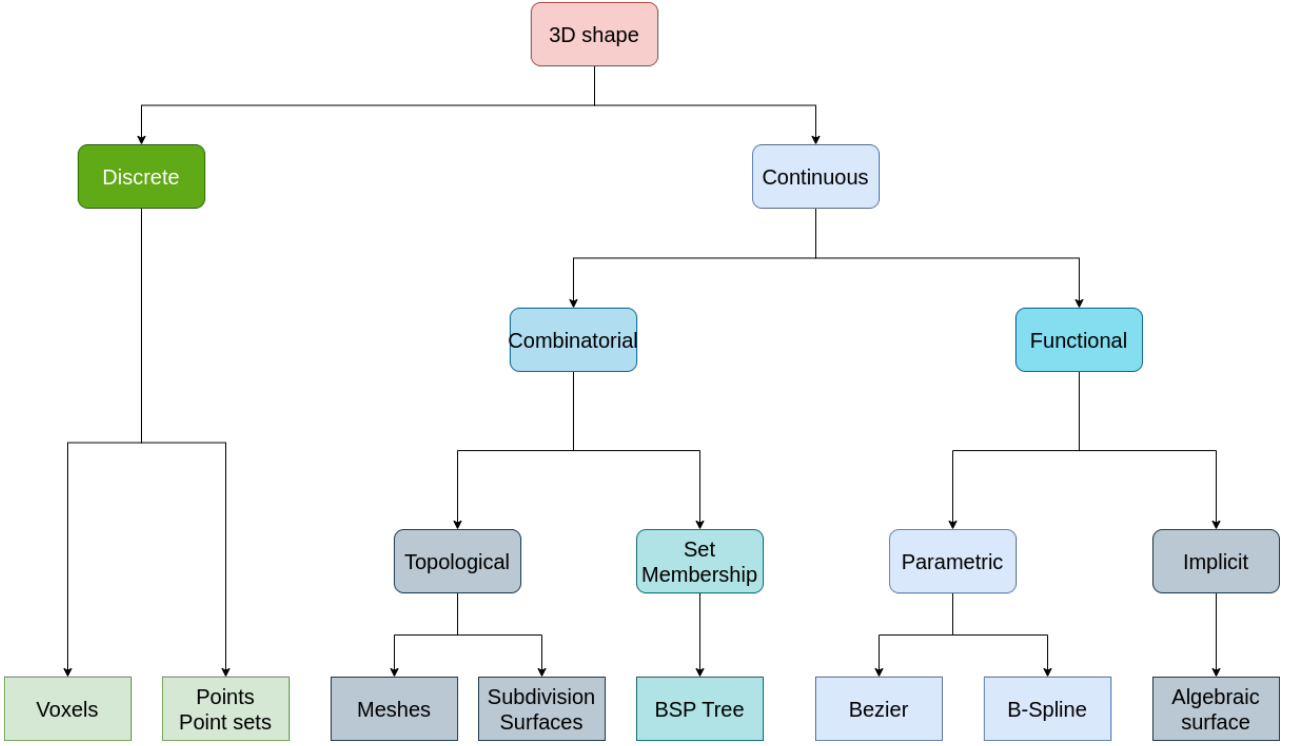




https://research.dreamworks.com/research_library/cloud-modeling-and-rendering-for-puss-in-boots/

Object shape representations

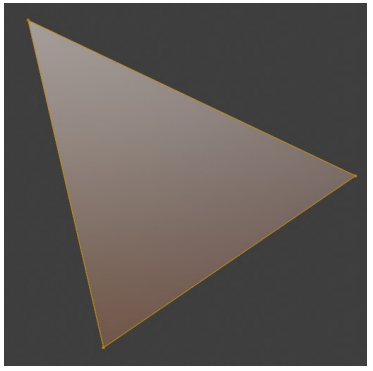
- **Points**
  - Point clouds
  - Particles and Particle systems
- **Surfaces**:
  - Polygonal mesh
  - Subdivision surfaces
  - Parametric surfaces
  - Implicit surfaces
- **Volumetric objects/solids**
  - Voxels
  - Space partitioning data-structures
- **High-level structures**
  - Scene graph

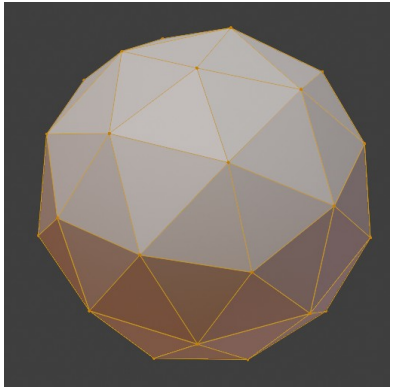


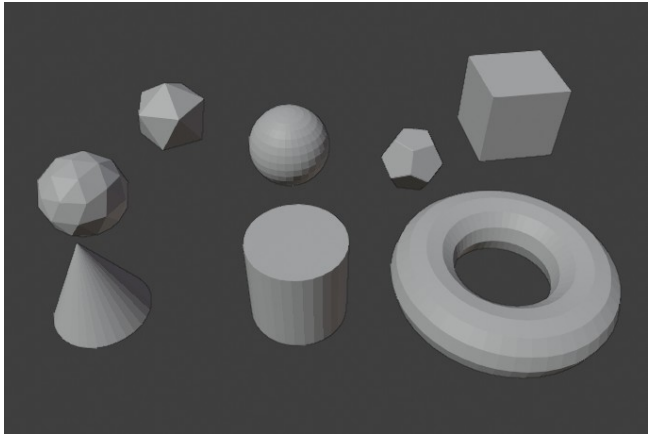
Based on:
https://www.cs.princeton.edu/courses/archive/spring22/cos426/lectures/Lecture-5.pdf

Shape representation 1: triangles

- Simplest way to define surface is using co-planar polygons with three points: **triangles**
  - Widely used surface **shape representation primitive** (especially in game engines, e.g. Unity)
  - It is **approximation primitive**: curved surfaces are approximated with triangles
  - Very **simple**, great properties for **easy calculation**, often used for **efficient rendering** purposes
  - Not suitable for modeling, thus different shape representations are introduced
  - For rendering purposes, often all representations are turned to triangles before/during rendering stage - using very elaborated method called **triangulation**

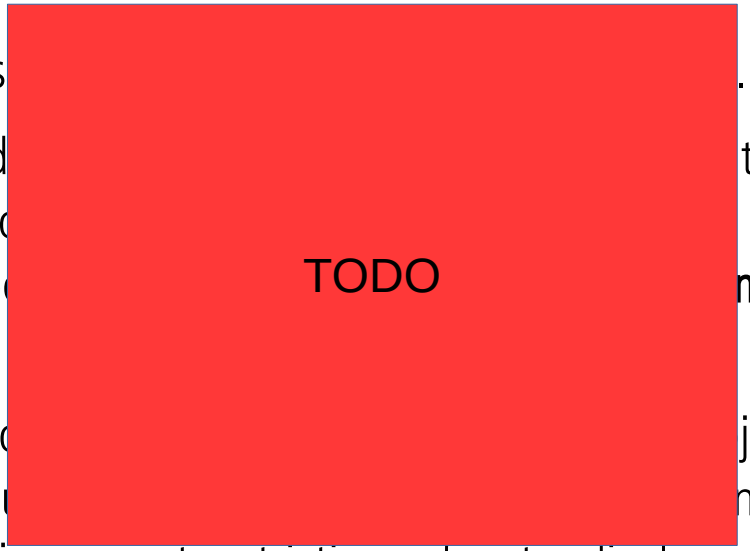






Shape representation 1: triangles

- Some shapes do not have flat surfaces! Triangle repres .
- If we would like to represent curved surface, we would the curved surface better. In this process we are placing mo Generally, converting smooth surface to triangulated po **mesh** – is called **tessellation**.
- Main point to take away is that we can represent any o jects will never be perfect representations of a real world, bu n be small enough to display objects in high quality taking in account restrictions of raster display.
  - Amount of details increase realistic representation but also complexity of rendering. Computer graphics often deals with trade-off between visual quality and speed**.

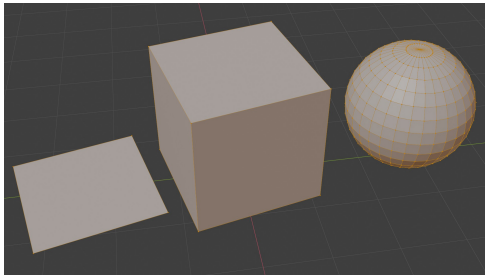


* There are methods in rendering which make surface looks smooth although is represented using polygons. This method is called smooth shading and we will discuss it later.

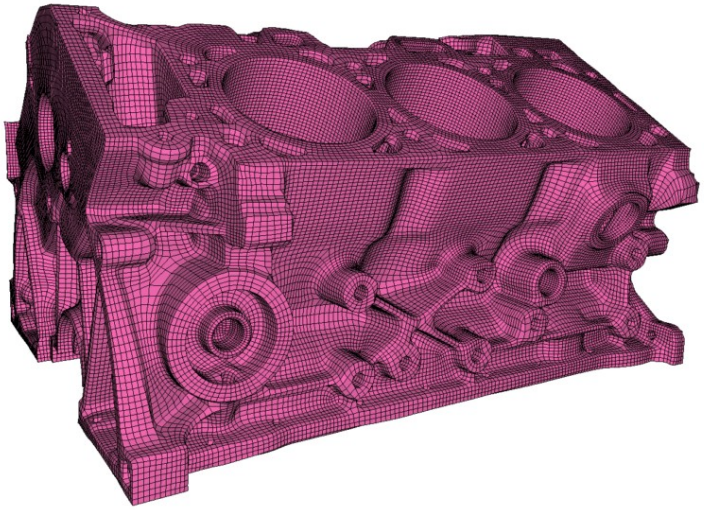
** Finding good tradeoff between visual quality and object complexity is big research field in computer graphics. Note also that distance of viewing plays important role in amount of detail that it has to be represented.

Shape representation 1: quad mesh

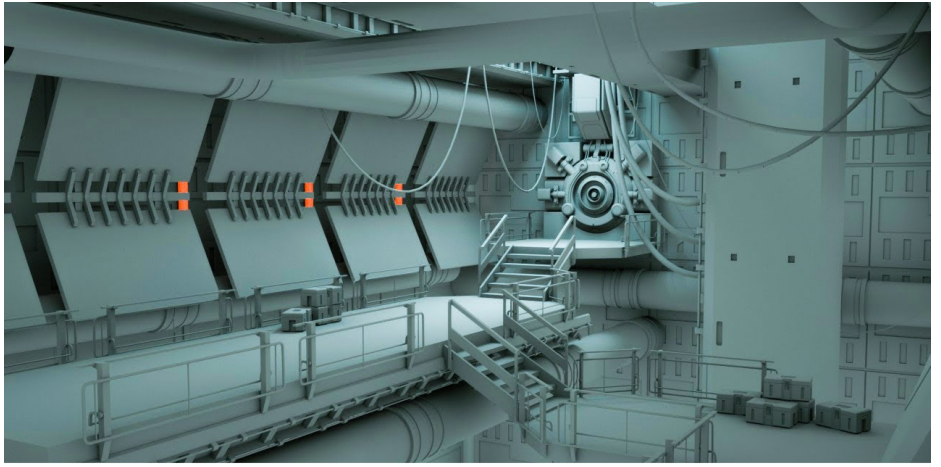
- Polygons with four vertices, not necessary co-planar
- Often provided by professional 3D modeling tools
- Often used for modeling (especially hard-surface modeling)



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



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

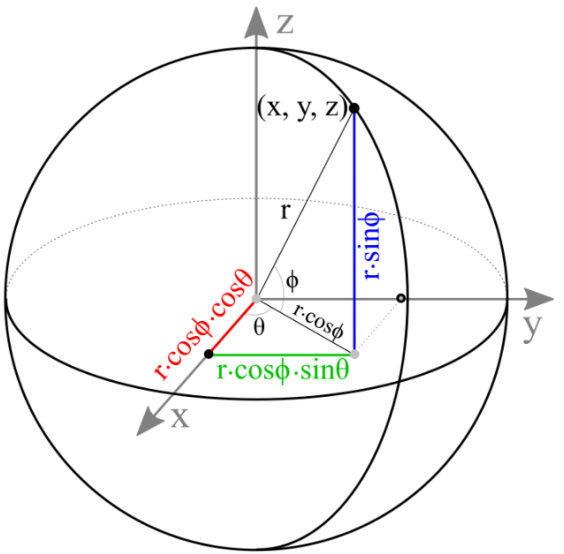


Hard surface modeling:
https://www.sidefx.com/tutorials/houdini-modelling-tutorial-175-houdini-hard-surface-modelling-tutorial/
Hard surface modeling: https://www.youtube.com/watch?v=1qVbGr_ie30&ab_channel=JoshGambrell

Shape representation 2: 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:

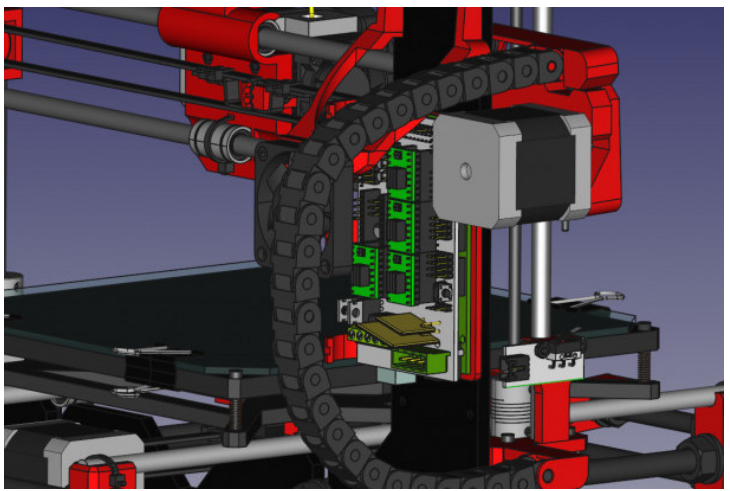
$$
\begin{aligned}
x &= (r \cdot \cos\phi) \cdot \cos\theta \\
y &= (r \cdot \cos\phi) \cdot \sin\theta \\
z &= r \cdot \sin\phi
\end{aligned}
$$



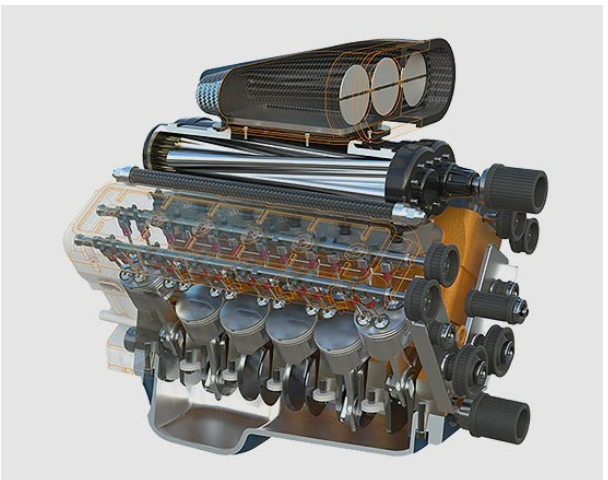
More on spheres in OpenGL:
http://www.songho.ca/opengl/gl_sphere.html

Shape representation 2: parametric surfaces

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



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

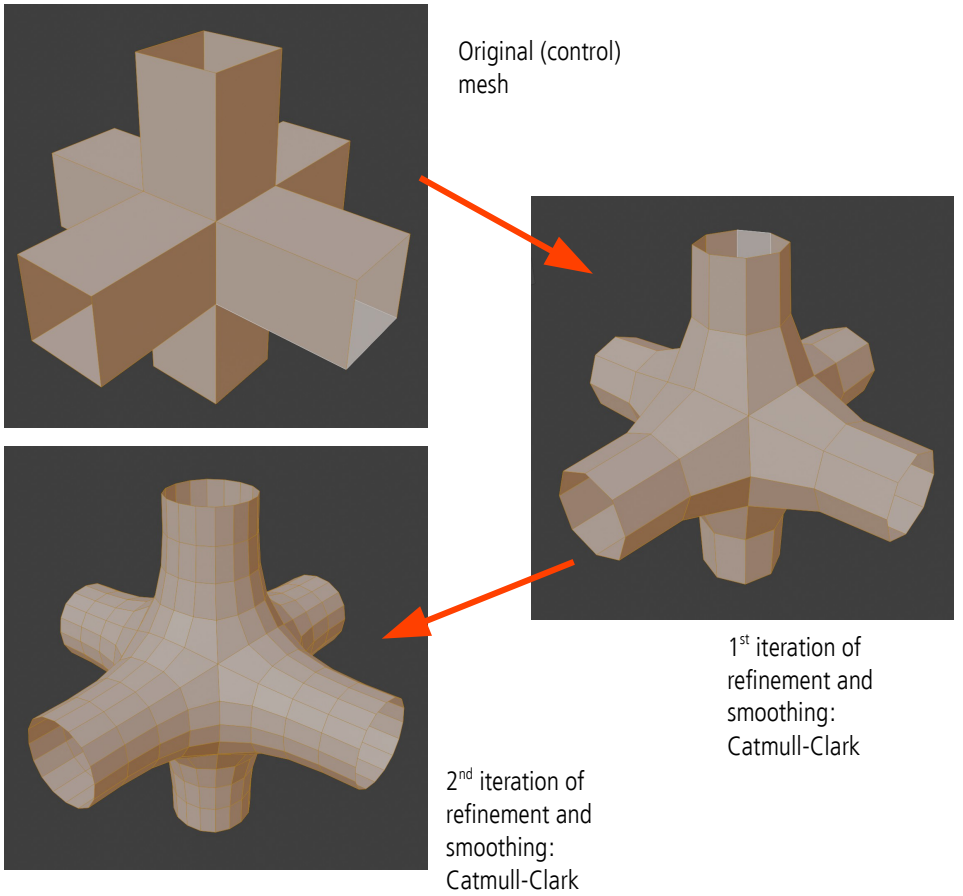


Autodesk Fusion 360: https://www.autodesk.com/products/fusion-360/overview

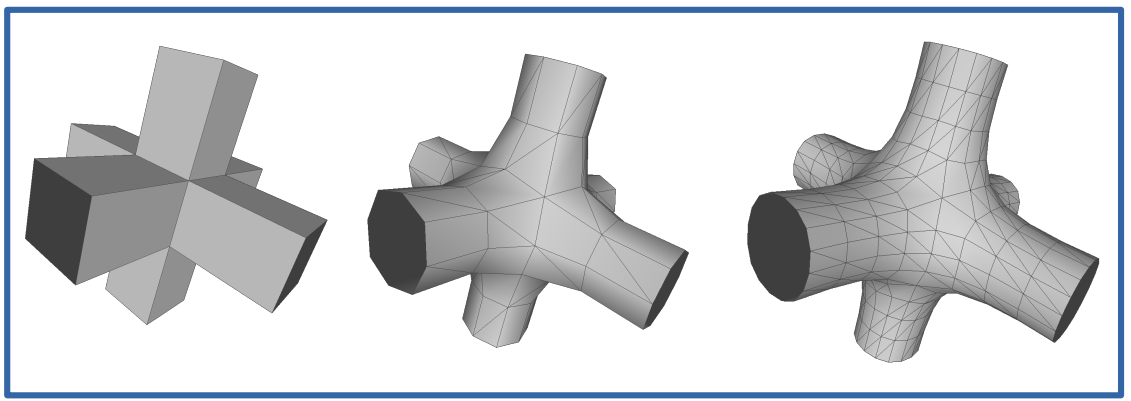
Renderman NURBS:
https://rmanwiki.pixar.com/display/REN24/NURBS

Shape representation 3: subdivision surfaces

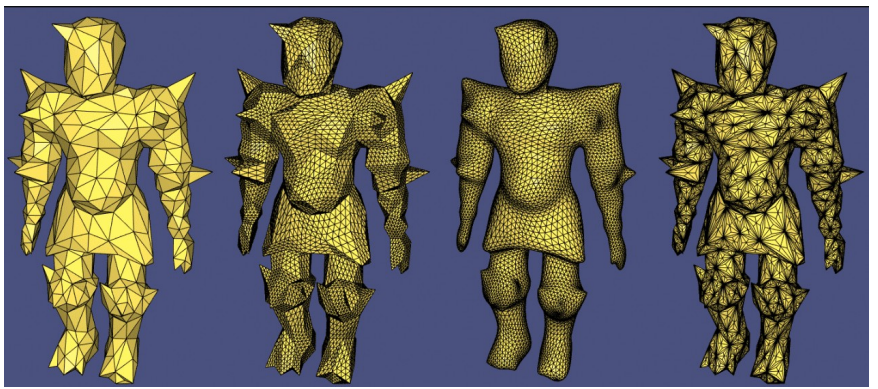
- Paradigm for defining smooth and continuous surfaces from meshes
- Provides infinite level of detail: as many triangles as needed can be generated. Steps:
  - Process starts with polygonal mesh called **control mesh** (or control cage)
  - First, **refinement phase**, creates new vertices and reconnects to create new smaller polygons
  - Second, **smoothing phase**, computes new positions for some or all vertices in the mesh
- Different subdivision schemes exist depending on methods used in refinement phase and smoothing phase. Approaches:
  - Loop's method, Catmull-Clark subidivison, etc.



Shape representation 3: subdivision surfaces



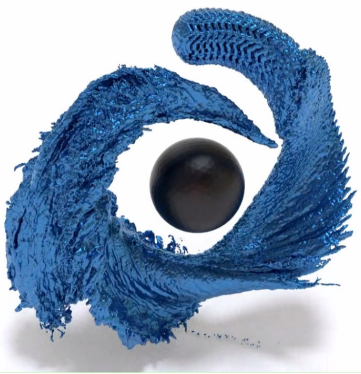
Meshlab loop subdivision:
https://pymeshlab.readthedocs.io/en/latest/filter_list.html#meshing_surface_subdivision_loop
https://github.com/cnr-isti-vclab/meshlab



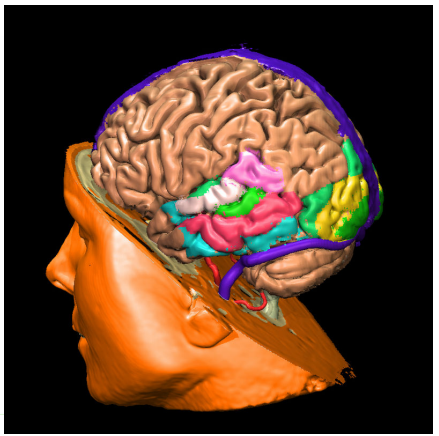
Libigl subdivision: https://libigl.github.io/tutorial/ (original, upsample, loop and barycentric.

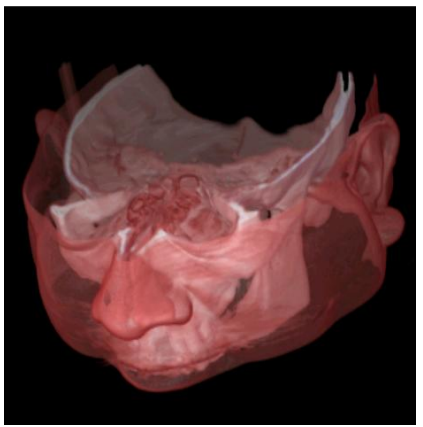
Shape representation 4: voxels

- **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

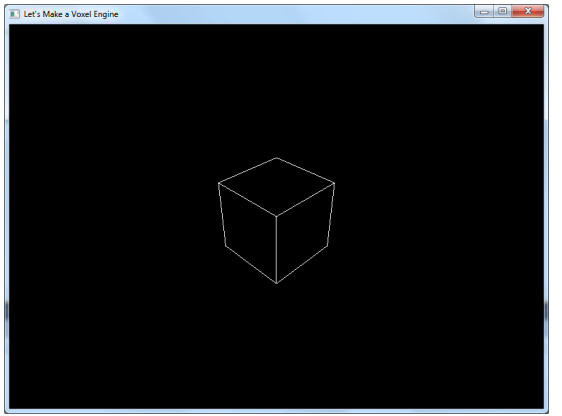


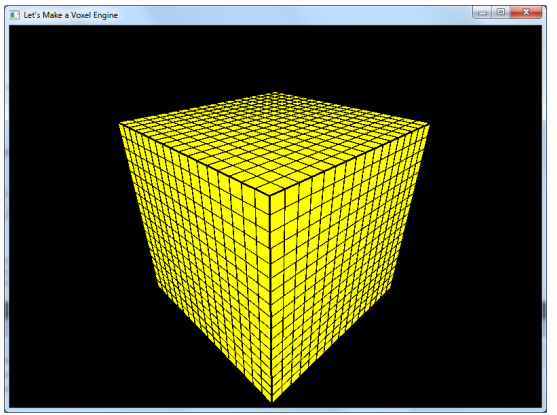
https://github.com/rlguy/GridFluidSim3D





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 4: voxels

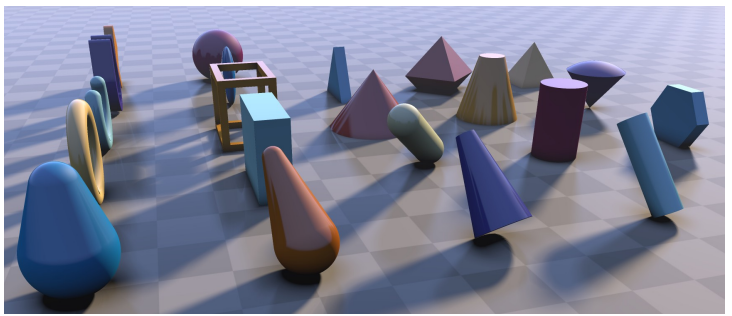
Voxel-based modeling: https://ephtracy.github.io/

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: implicit

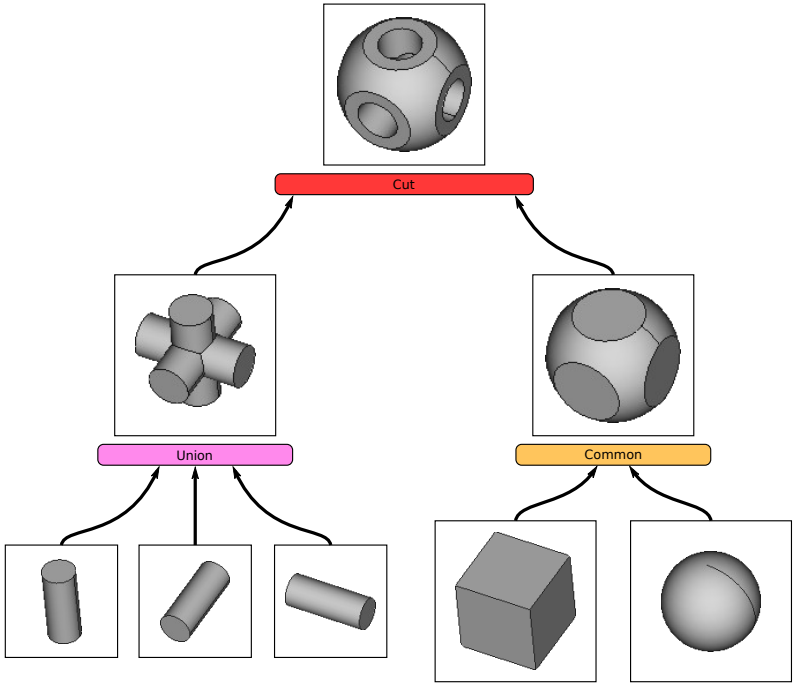
- Point on surface is described with **implicit – signed distance – function (SDF):** `f(x,y,z) = f(p) = 0`
  - Point is on surface if `f(p) = 0`. If `f(p) < 0` then `p` inside, otherwise `p` is outside
- **Simple shapes** can be defined using SDFs: cubes, spheres, etc.
  - Example: sphere centered at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



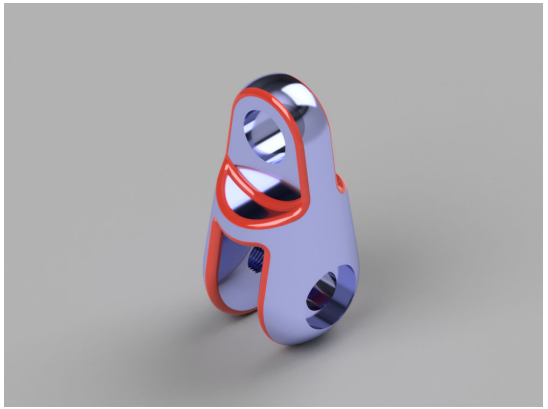
Implicit shape representations:
https://iquilezles.org/articles/distfunctions/
https://www.youtube.com/watch?v=62-pRVZuS5c&ab_channel=InigoQuilez

Shape representation 5: 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

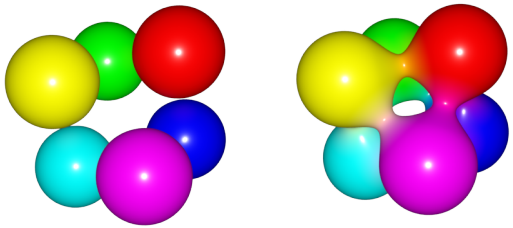


https://wiki.freecadweb.org/Constructive_solid_geometry



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

Shape representation 5: implicit

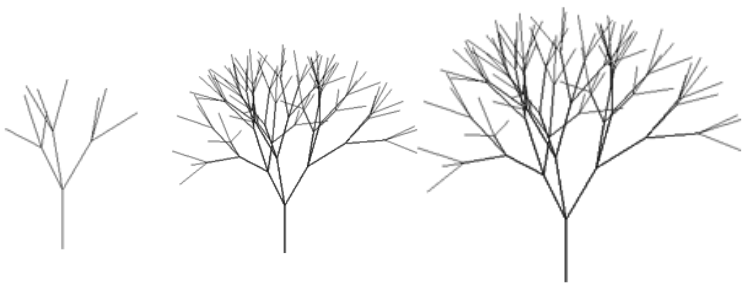


Good for representing organic objects (metaballs):
https://rmanwiki.pixar.com/display/REN24/Implicit+Surfaces
https://docs.blender.org/manual/en/latest/modeling/metas/introduction.html

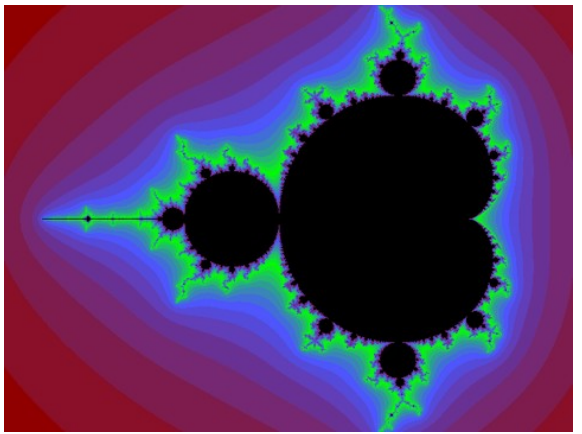
Implicit representation for modeling and rendering:
https://www.gdcvault.com/play/1025316/Advanced-Graphics-Techniques-Tutorial-GPU

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/

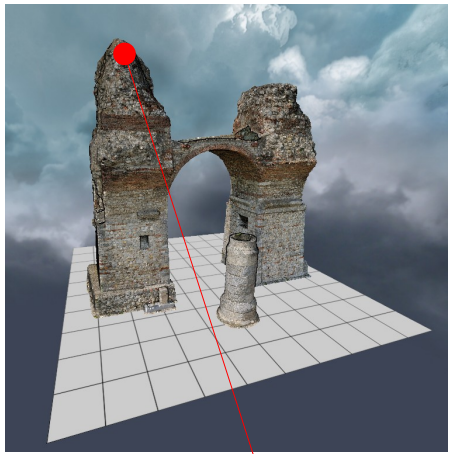


L-Systems: grammar for complex shapes generation using iterations:
https://www.sidefx.com/docs/houdini/nodes/sop/lsystem.html
https://www.youtube.com/watch?v=0vE8GiXhOWM&t=1248s&ab_channel=Houdini
https://www.jonathanrpace.com/project_trees.html

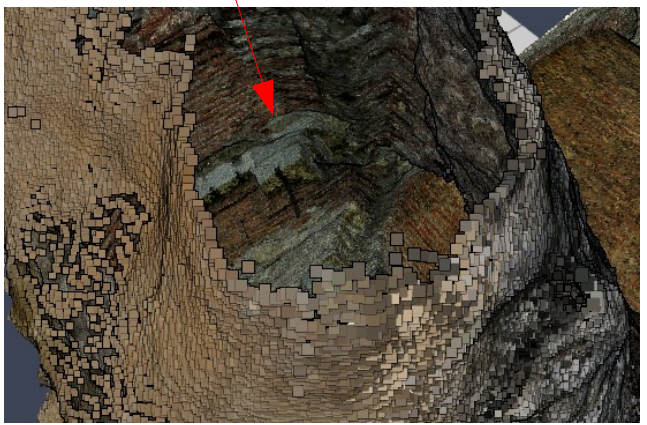


Mandelbrot: https://www.mandelbulb.com/
https://www.geeks3d.com/20091116/exploration-of-the-real-3d-mandelbrot-fractal/

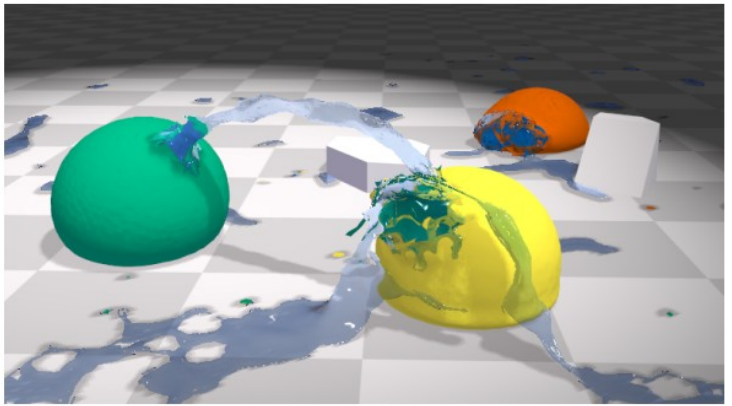
Shape representation 8: Points

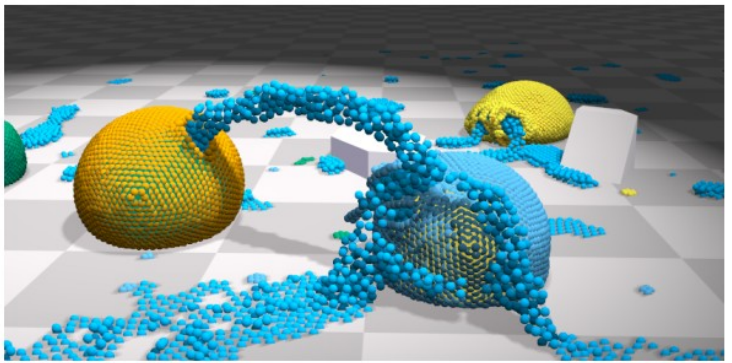


Scanned real-world objects can be stored as **point clouds**



https://github.com/potree/potree



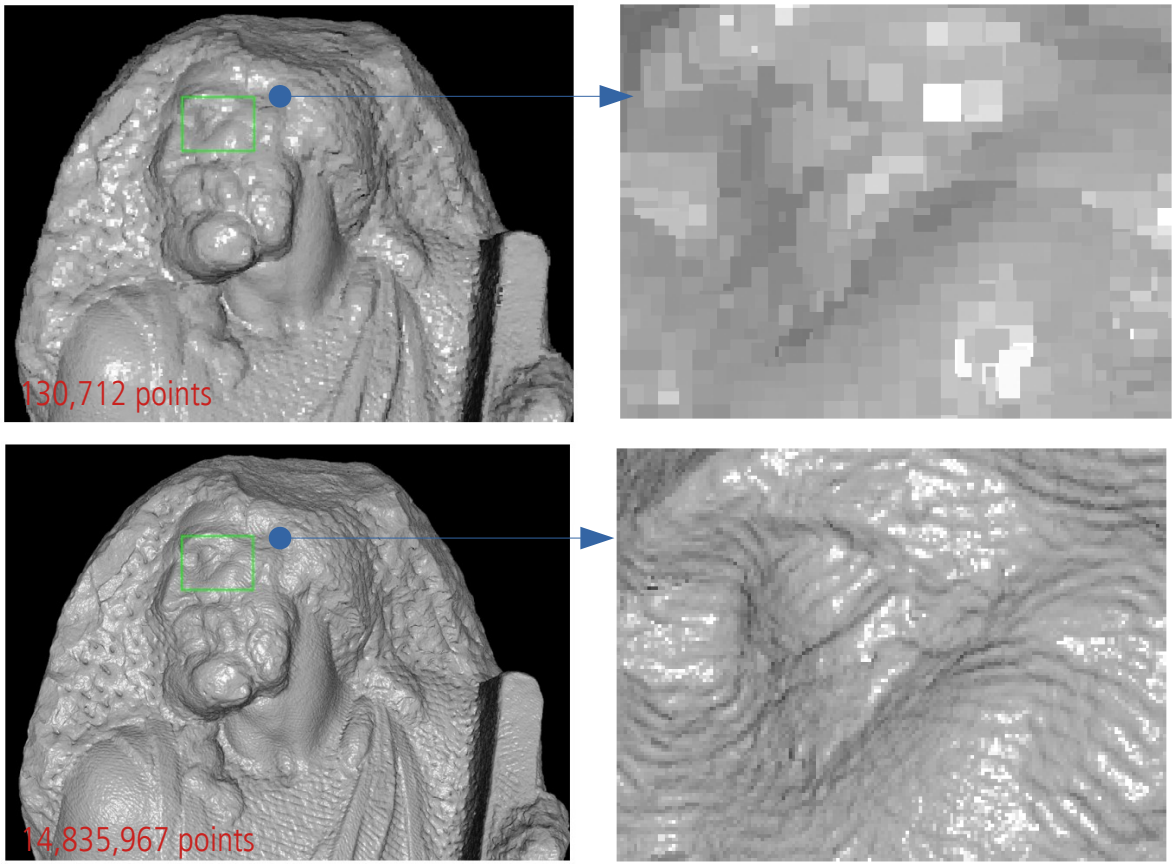


Unified physics for real-time applications using points:
https://mmacklin.com/uppfrta_preprint.pdf

Shape representation 8: 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

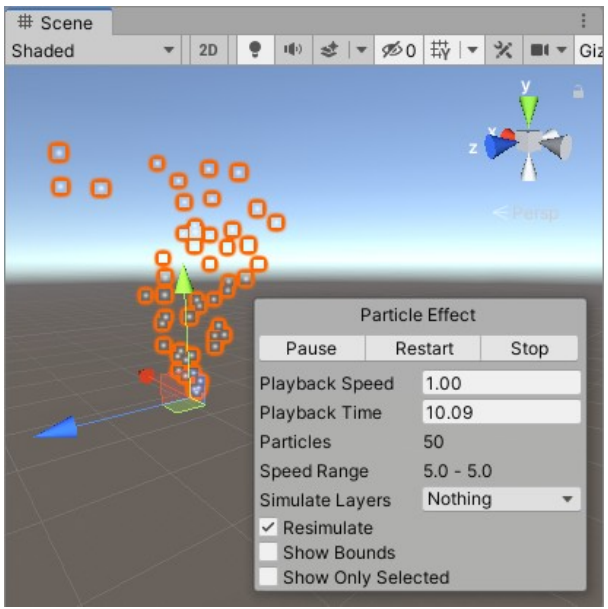


Qsplat: http://graphics.stanford.edu/software/qsplat/

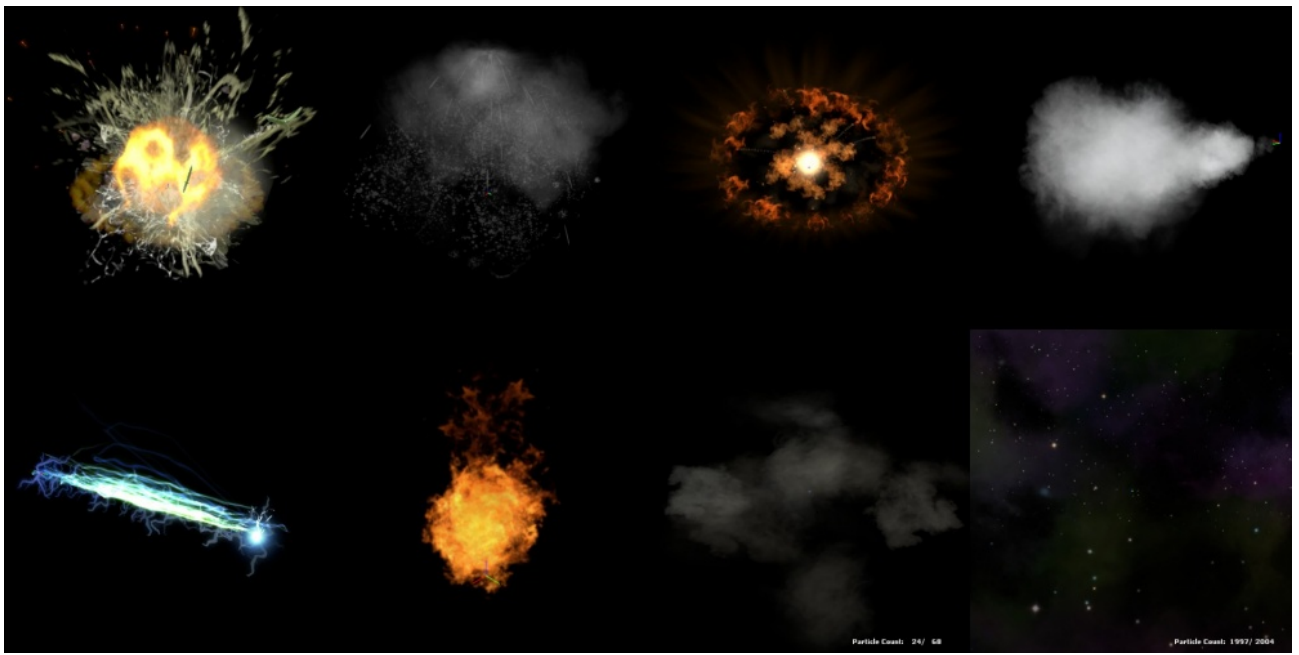
* Surfels:
https://vcg.seas.harvard.edu/publications/surfels-surface-elements-as-rendering-primitives

Shape representation 9: 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.



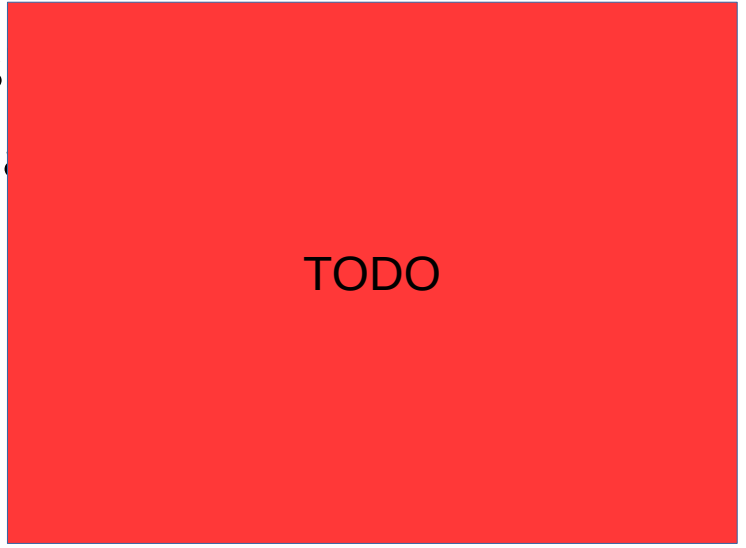
https://docs.unity3d.com/ScriptReference/ParticleSystem.html



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

Equivalence of representations

- Each shape representation has capacity to describe s
- Different representations exist because certain tasks
  representation. For example:
  - Rendering process
  - 3D modeling
  - Simulation of objects
  - Animation
  - Acquisition/digitalization of objects from a real world
- Different datastructures used for representations determine algorithms for processing



3D models: various representations?

Similar as in, e.g., drawing where drawing tools and shapes ne based on set of representations and operations from which we must one hand flexible enough to be used for any shape and understan ram:

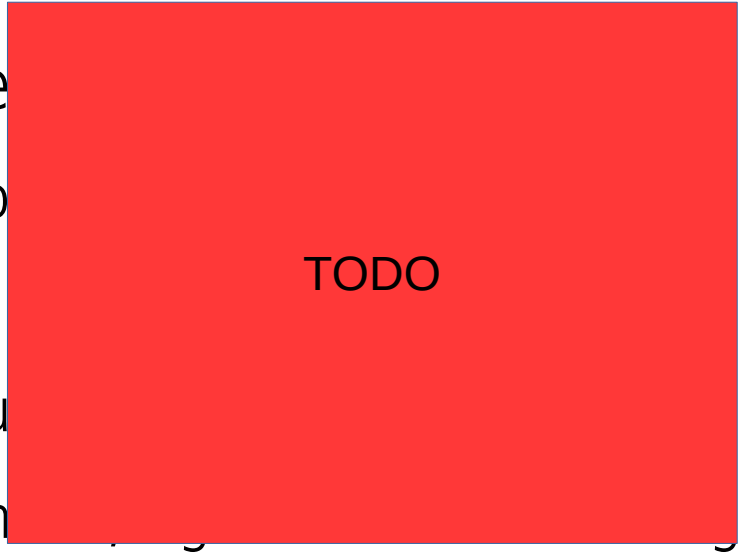
- From the user-creation point of side: we need a representa ve to handle (to author)
- From the rendering point of side: we need to make sure tha be used for rendering process
  - Shape representation will be used for determining visibility proble : what objects we see from certain point of view
  - Material representation will be used for determining the shading problem**: how the objects that we see appear

* As we will get to know more complex material representations we will see that this is not completely true. Visibility of objects will also depend on material. Imagine glass cup on the top of a book – book will be visible since glass is transparent.

** Shape and material information are both important for calculating how objects appear. We separate them to show that you can first purely focus on modeling a 3D shape and then model material afterwords. Also, separating objects in shape and material is desired for rendering architectures so that are decoupled.

3D models: various representations?

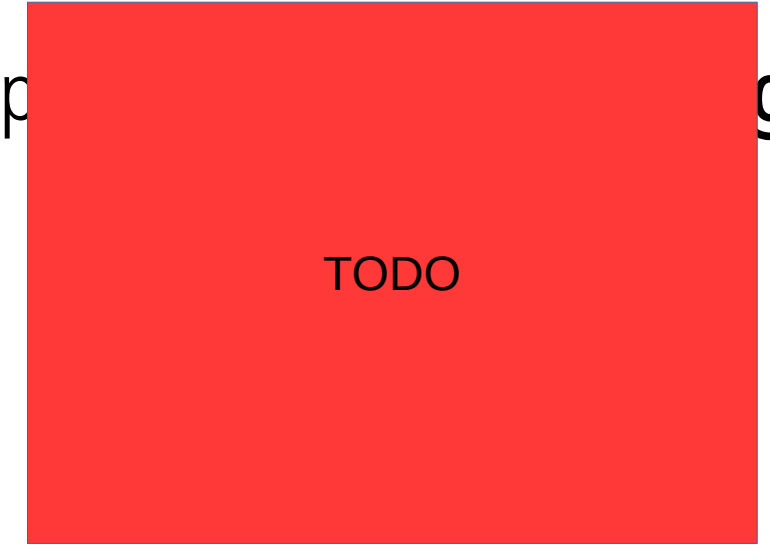
- Different tasks require different repre
  - Reconstructing from photographs: pho
  - Creating isosurfaces from CAT or MRI
  - Transforming data into surfaces or volu
  - Sampling real world objects using scan devices
    - Example: kinect



3D models: various representations?

Different tasks require different rep g:

- Manual modeling
- Procedural modeling



3D models: various representations?

Different tasks require different representations – **modeling**:

- Sculpting

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

TODO

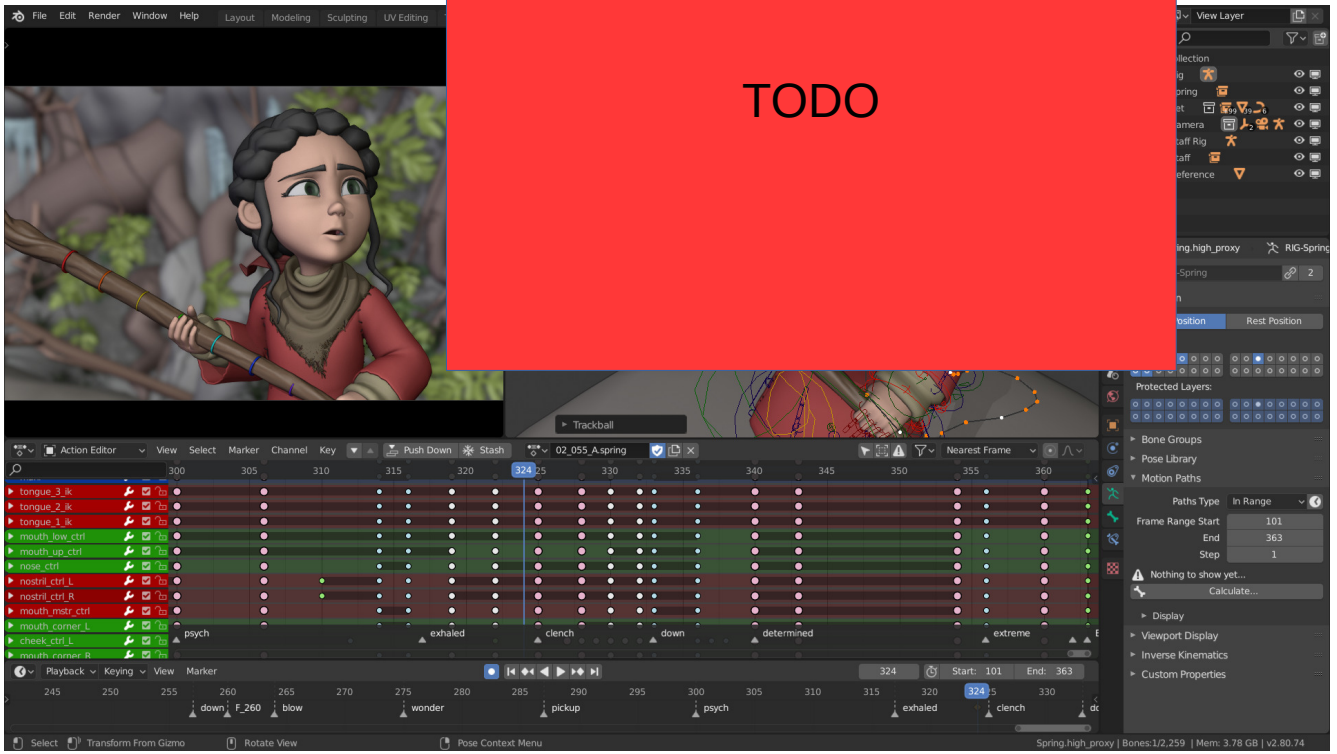
Blender, https://www.blender.org/features/sculpting/

3D models: various representations?

Different tasks require different repr

- Manual, rigging

TODO

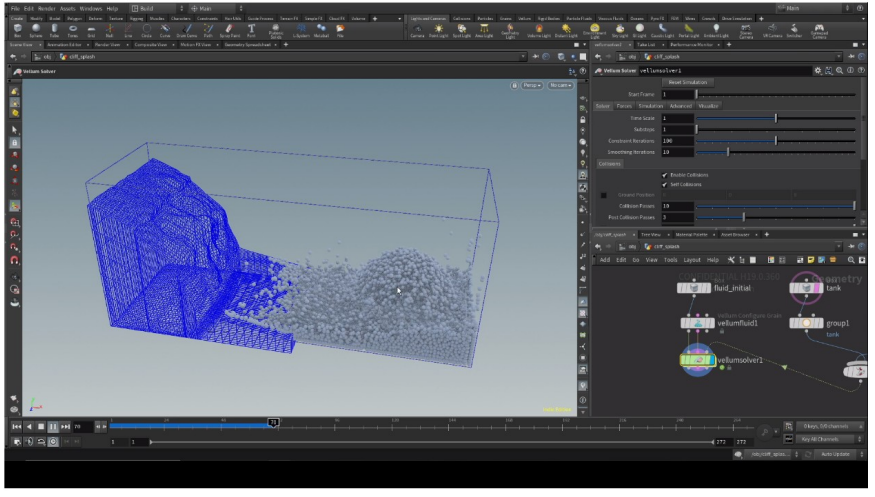


https://www.blender.org/features/animation/

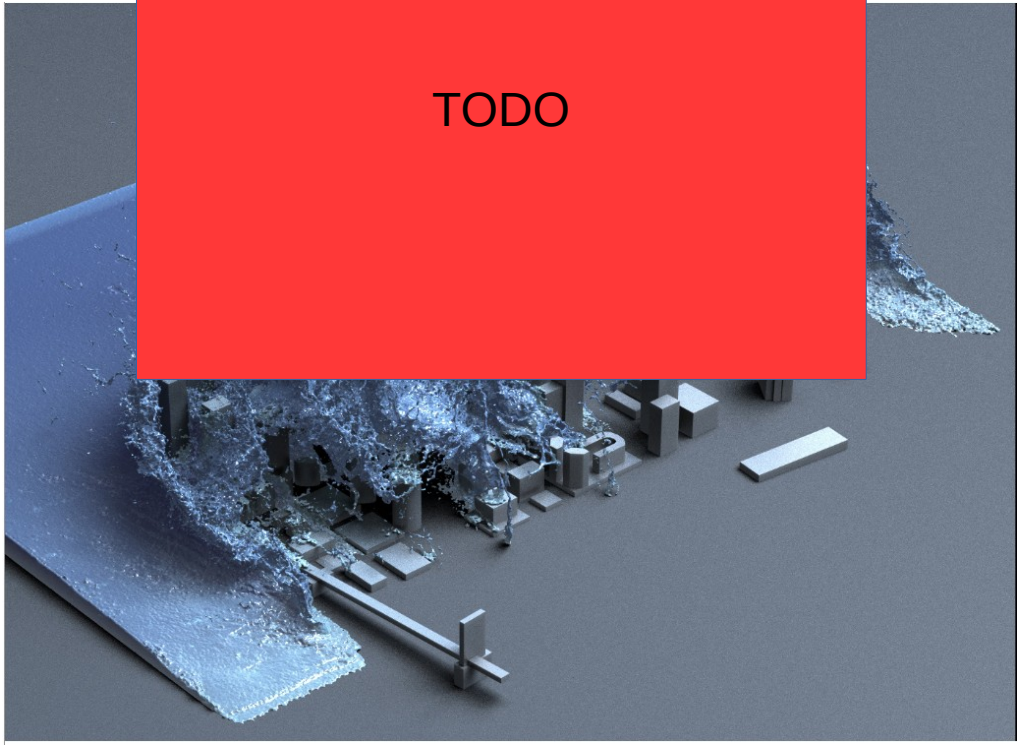
3D models: various representations?

Different tasks require different rep**on**:

- procedural, simulation



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

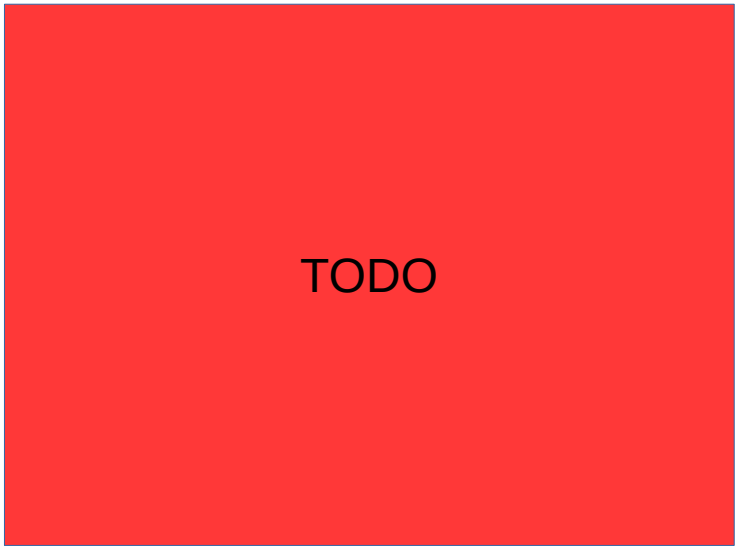


OpenVDB, https://people.csail.mit.edu/kuiwu/gvdb_sim.html

TODO

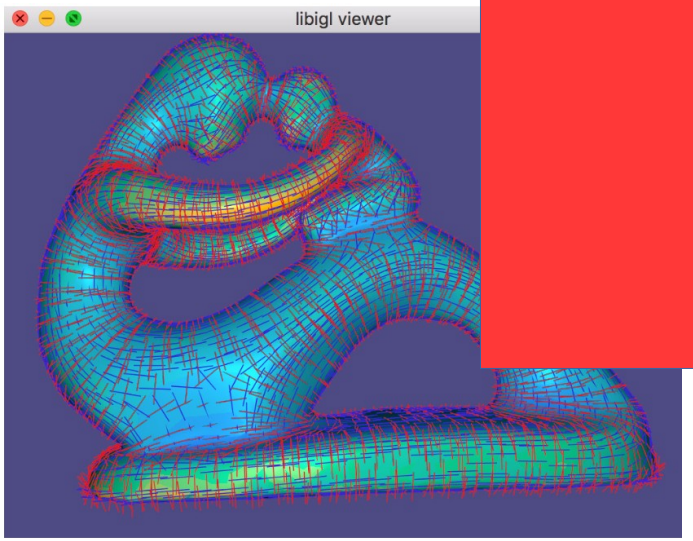
3D models: various representations?

- Different tasks require different **rendering**:
  - Efficient visibility solving

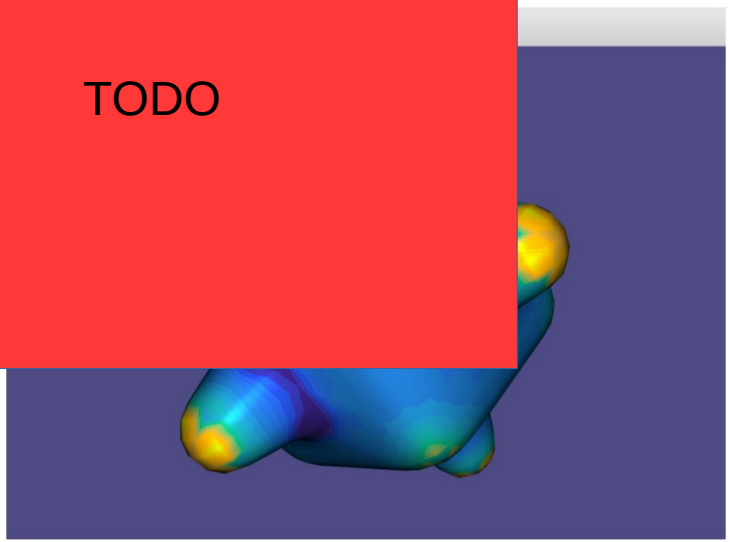


3D models: various representations?

- Different tasks require different **sis**:
  - Curvature
  - Smoothness



Libigl: principled curvature directions vectors

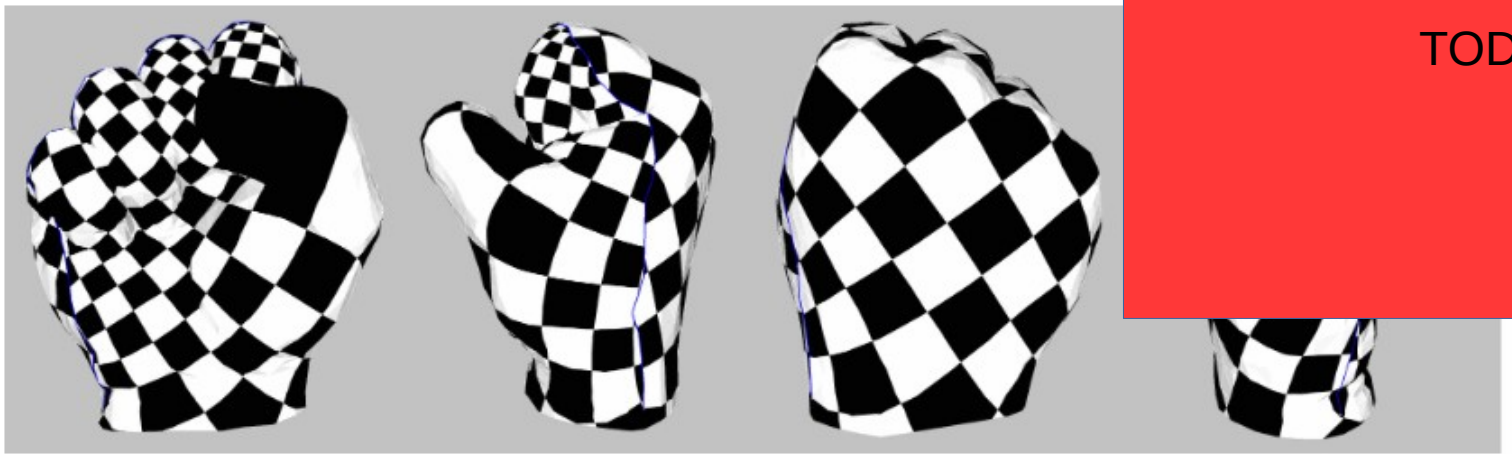


Libigl: Gaussian curvature visualization using pseudocolor

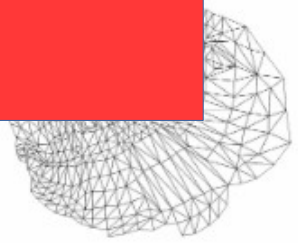
TODO

3D models: various representations?

- Different tasks require different representa
  - Parameterization



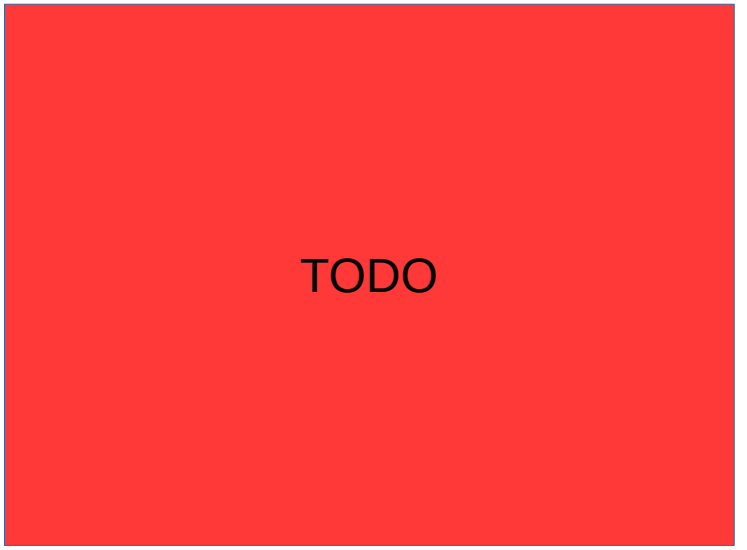
TODO



CGAL: Least Squares Conformal Maps

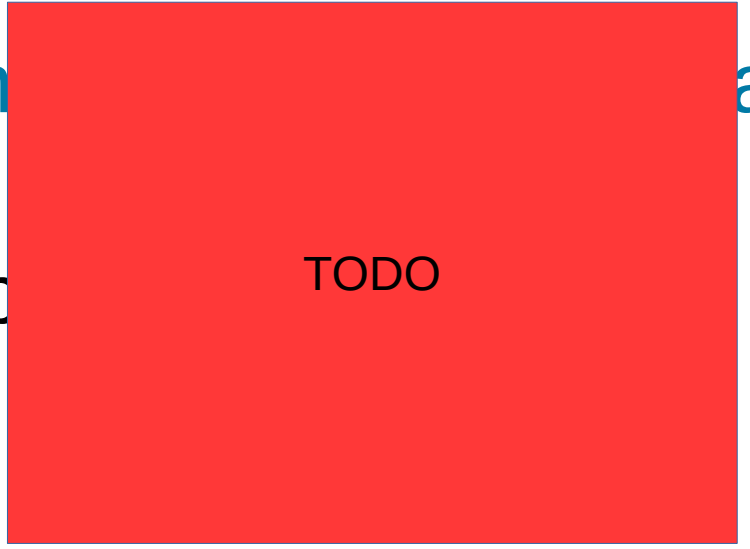
3D models: various representations?

- OTHER examples



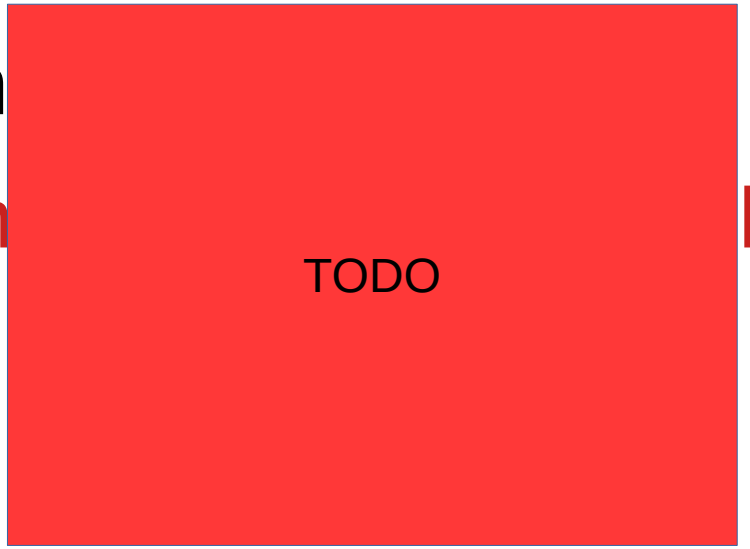
Materials

- https://rmanwiki.pixar.com adi ng
- How light interacts with ob
- TODO



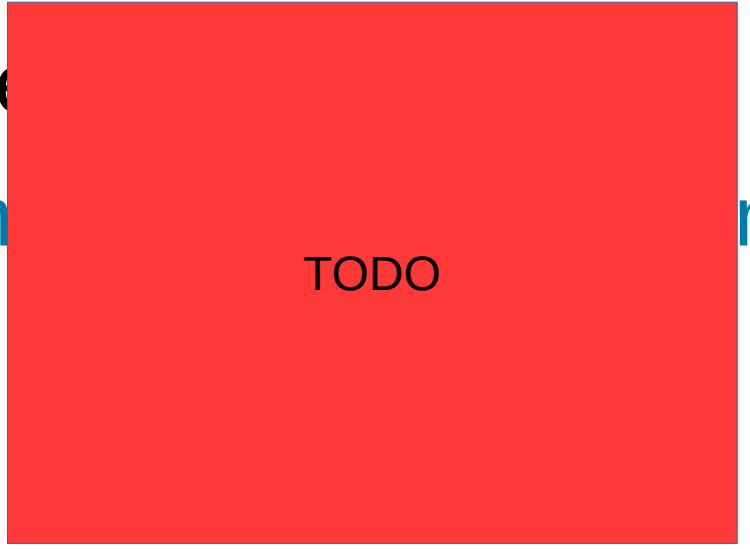
Lights

- How are light sources sim
- https://rmanwiki.pixar.com hti ng
-
- TODO



Cameras

- How are cameras simulate
- https://rmanwiki.pixar.com m eras
-
- TODO



Complex scene

<IMAGE: An motivation image that we will understand by the end of the lecture.>

Literature

- https://github.com/lorentzo/IntroductionToComputerGraphics/wiki/Foundations-of-3D-scene-modeling