01 - Introduction

Acknowledgements: Daniele Panozzo

Lecturer

Xifeng Gao

https://gaoxifeng.github.io/

g.gaoxifeng@gmail.com

Who Am I?

Course Goals

- Learn theory and applications of 3D mesh processing
- Learn how to design, program and analyze algorithms for **3D shape modeling** and **digital geometry processing**
- Hands-on experience with shape modeling and geometry processing algorithms

Prerequisites

- This course is suitable for both undergraduates and graduates, if you have the following prerequisites
- Courses:
    - Math, e.g., Calculus and Linear Algebra
    - Data Structure
- Git, github
- Familiar with C++ programming

Organization

- Course repository/website: https://github.com/FSU-ComputerGraphics/2022-3D-modeling-and-geometry-processing
- Communication through QQ Group
- Two weeks: July 19-22 and 25-28, 10:10AM -12AM
- Platform: VooV Meeting (Tencent Meeting)

Lectures

- I will upload the slides on the website before the class, so that you can directly annotate them
- You are encourage to take a look at the material **before** I present it in class

Lectures

- **Please interrupt me at any time to ask questions**

Grading

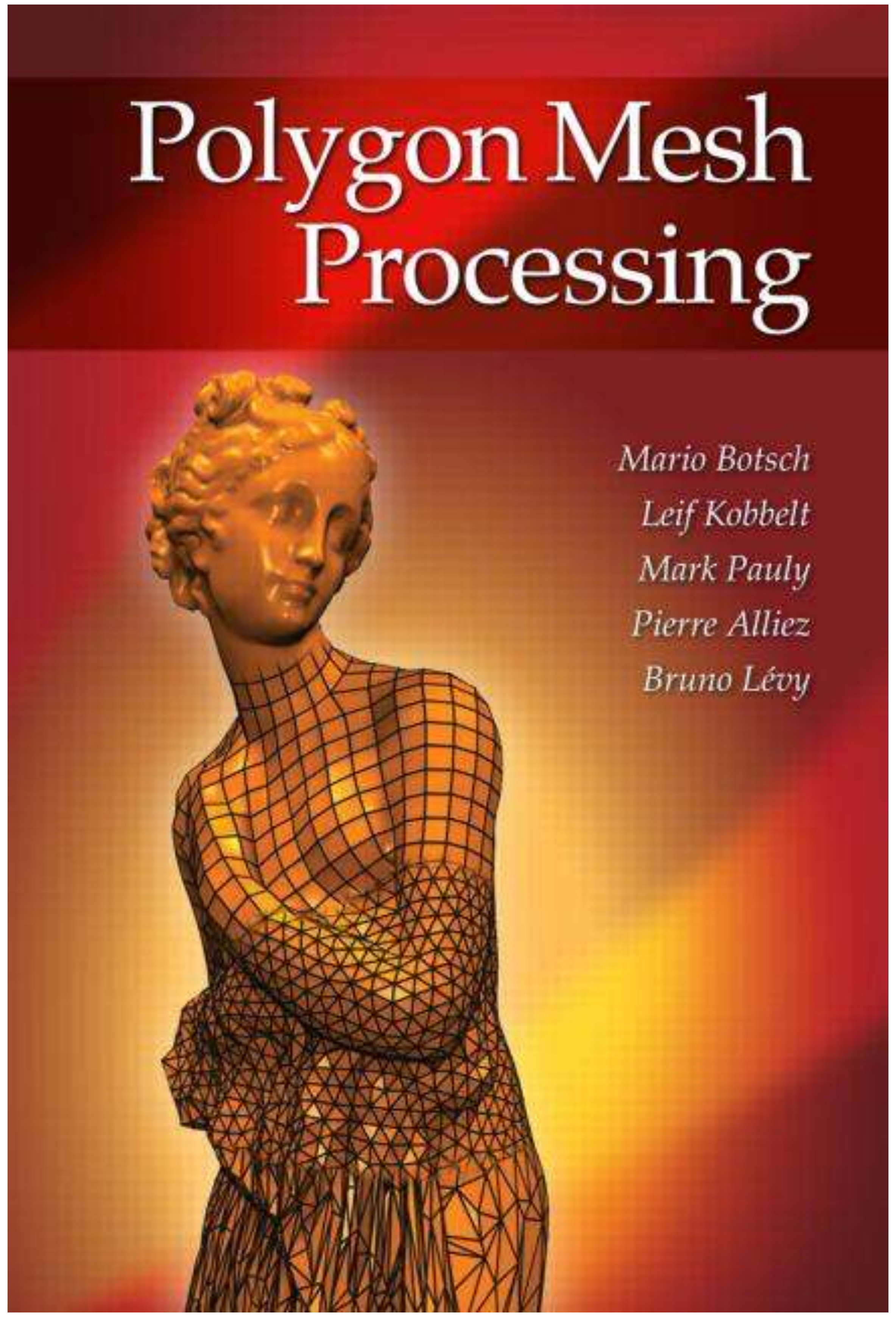
Tasks	Scores
Attendance	20
In-class tasks	20
1 project (code+report)	60

- You are encouraged to consult with your classmates/friends but collaboration in the assignments is **NOT** allowed.
- You are **NOT** allowed to copy code online or use external libraries (except those provided in the class).

Coding Project

- C++, or Python, choose the one you familiar with.
- Based on Libigl library: https://libigl.github.io/
- Submit To TA

Material

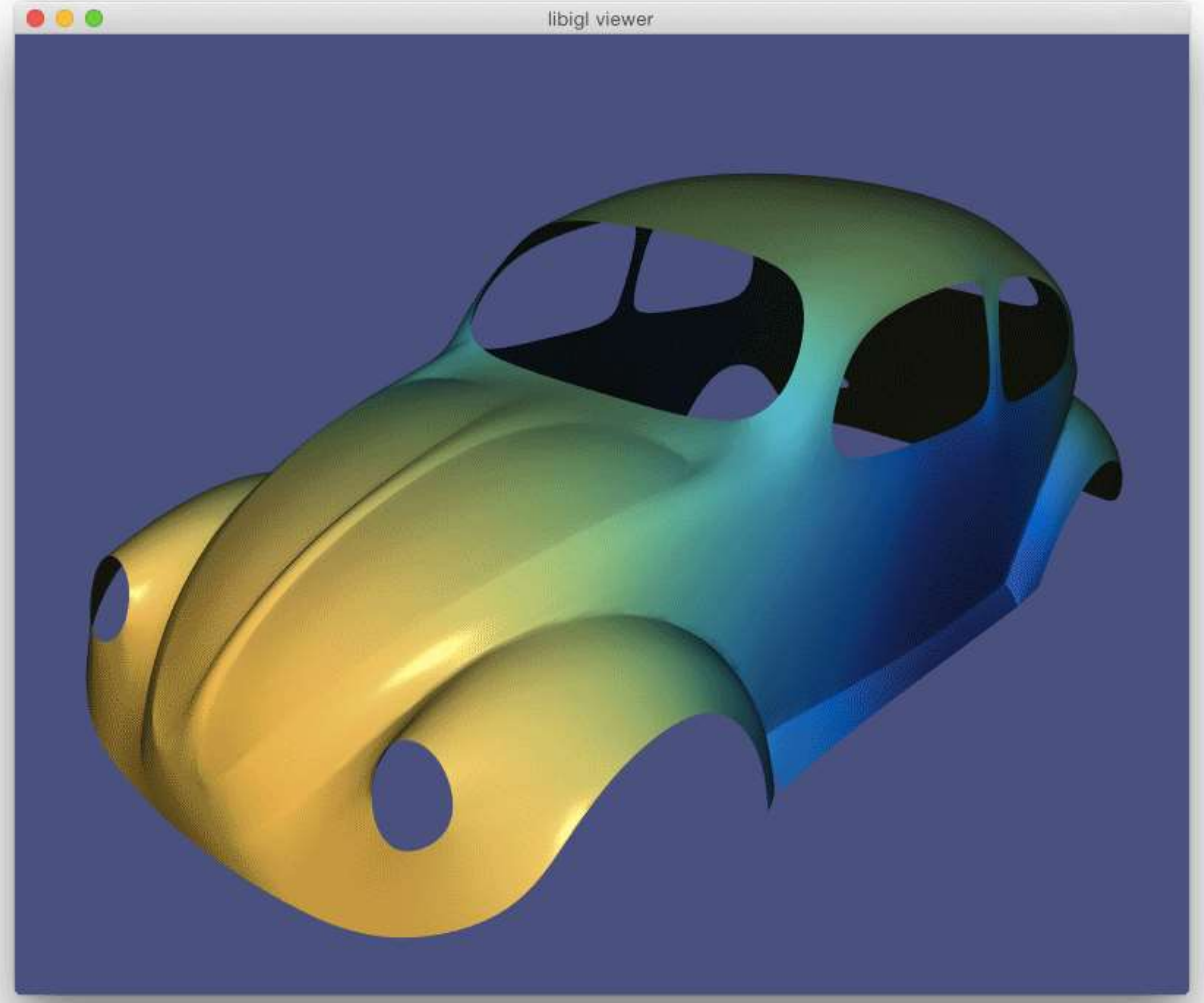


Polygon Mesh Processing

https://libigl.github.io/tutorial/

Geometric Modeling and Processing

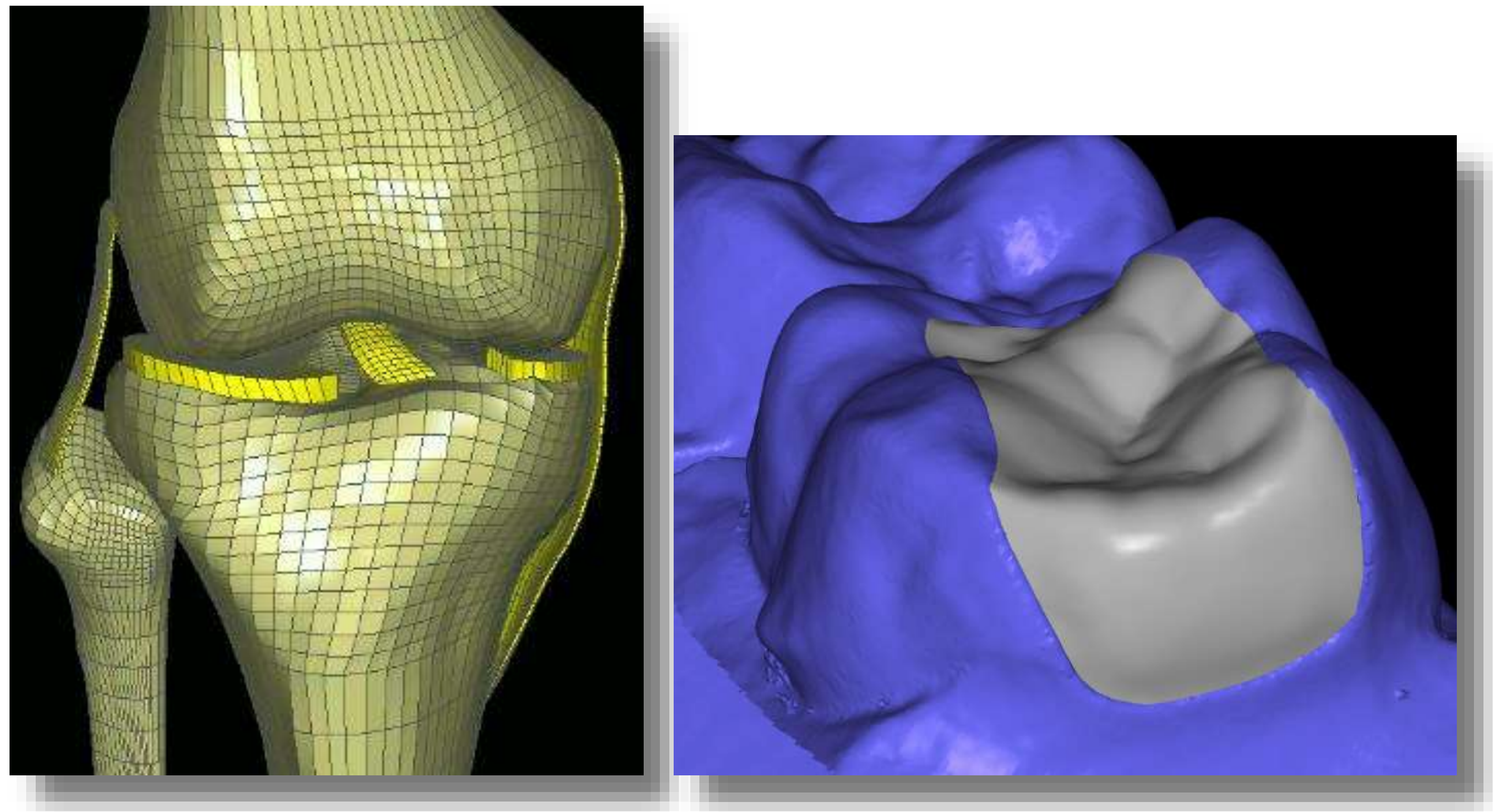
- The shape of an object is an important characteristic (not the only one…)



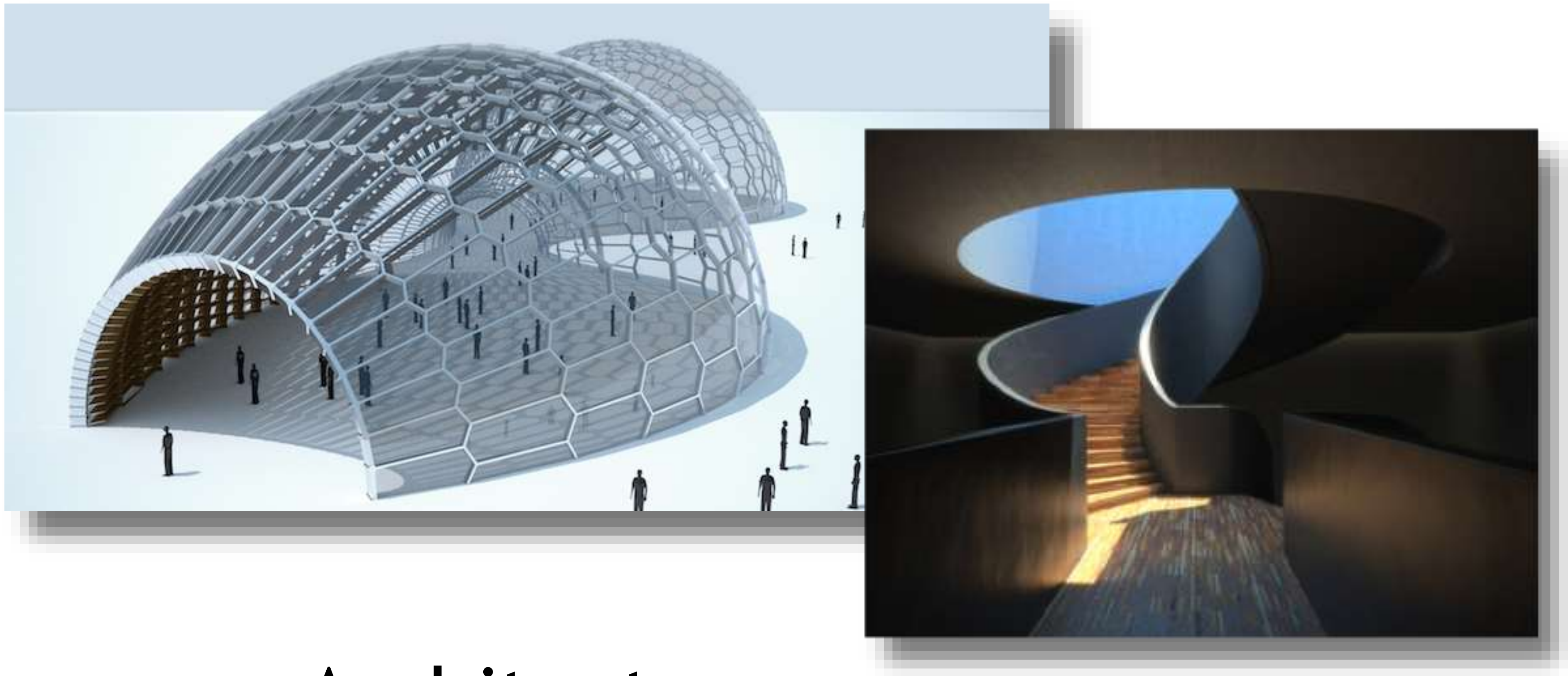
- Geometry processing: computerized modeling of 2D/3D geometry

Applications

Product design and prototyping



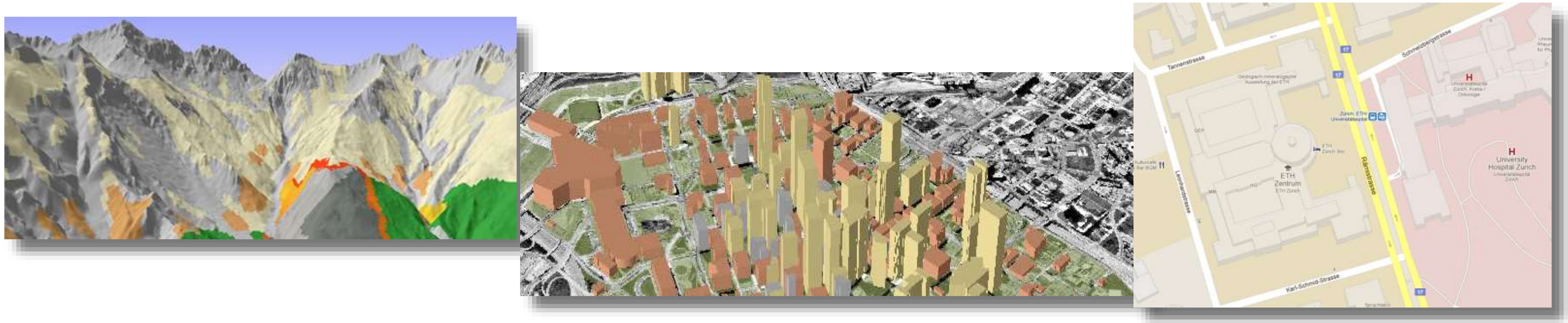
Medicine, prosthetics



Architecture

Cultural heritage

Applications



Geographical systems

[Bickel et al., ACM SIGGRAPH 2010]

Manufacturing, 3D Printing

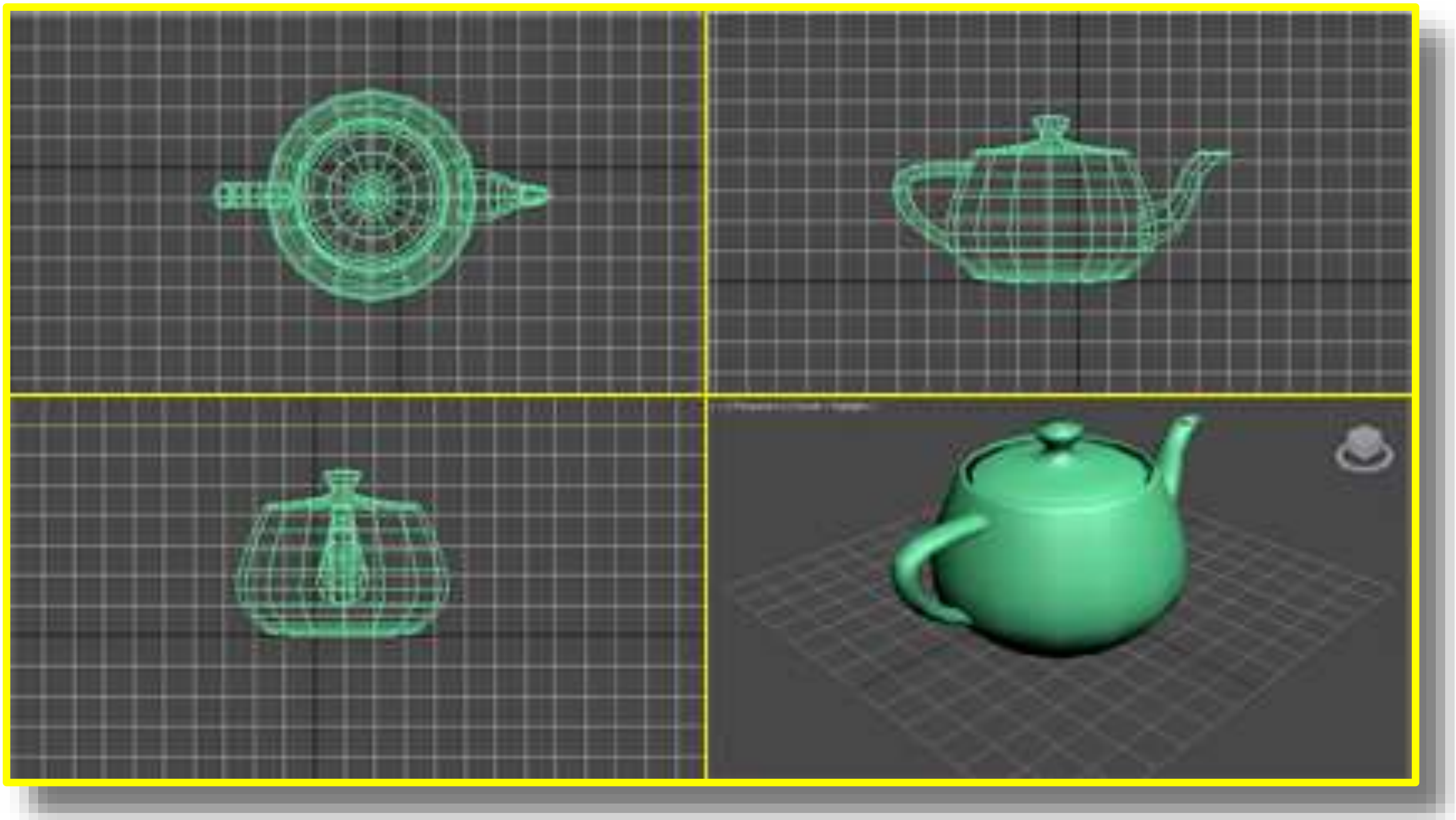
Applications

Applications

Digital Shape Modeling

- How do shapes find their way into computers?
  - Geometric modeling is difficult

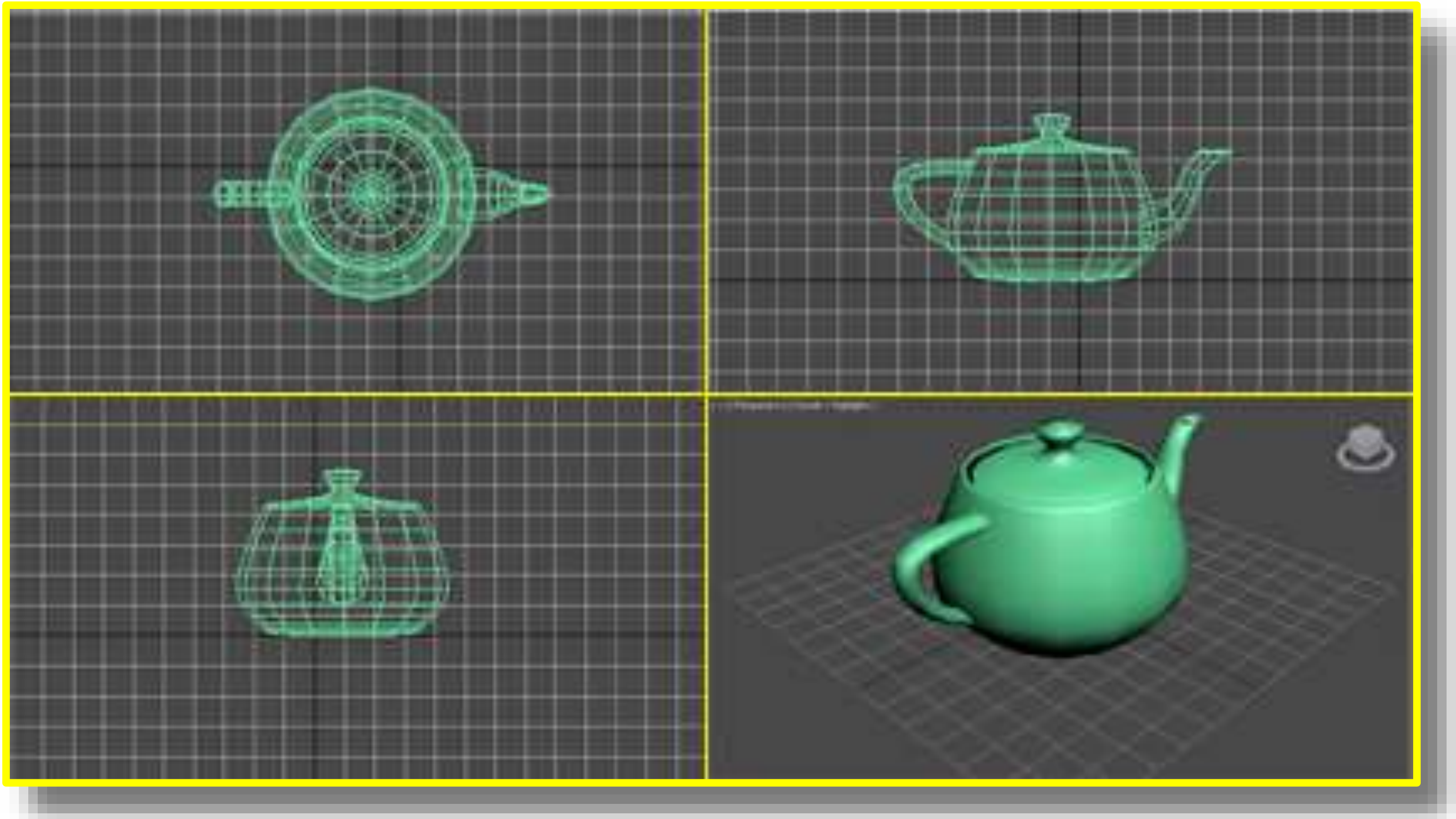
Humans have no
direct “video out”



“Translation” from
2D to 3D is hard

Digital Shape Modeling

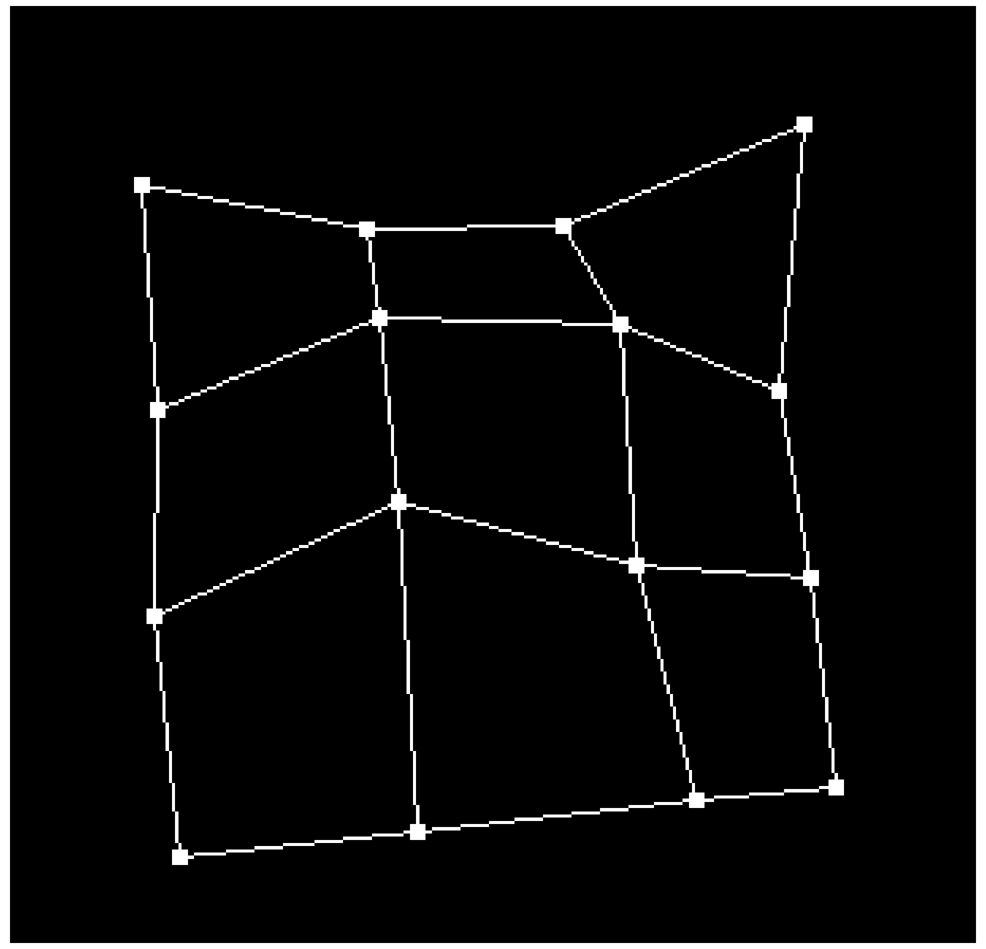
- How do shapes find their way into computers?
  - Geometric modeling is difficult



Use computation to compensate for lack of direct ability to convey visual information

Computer-Aided Geometric Design

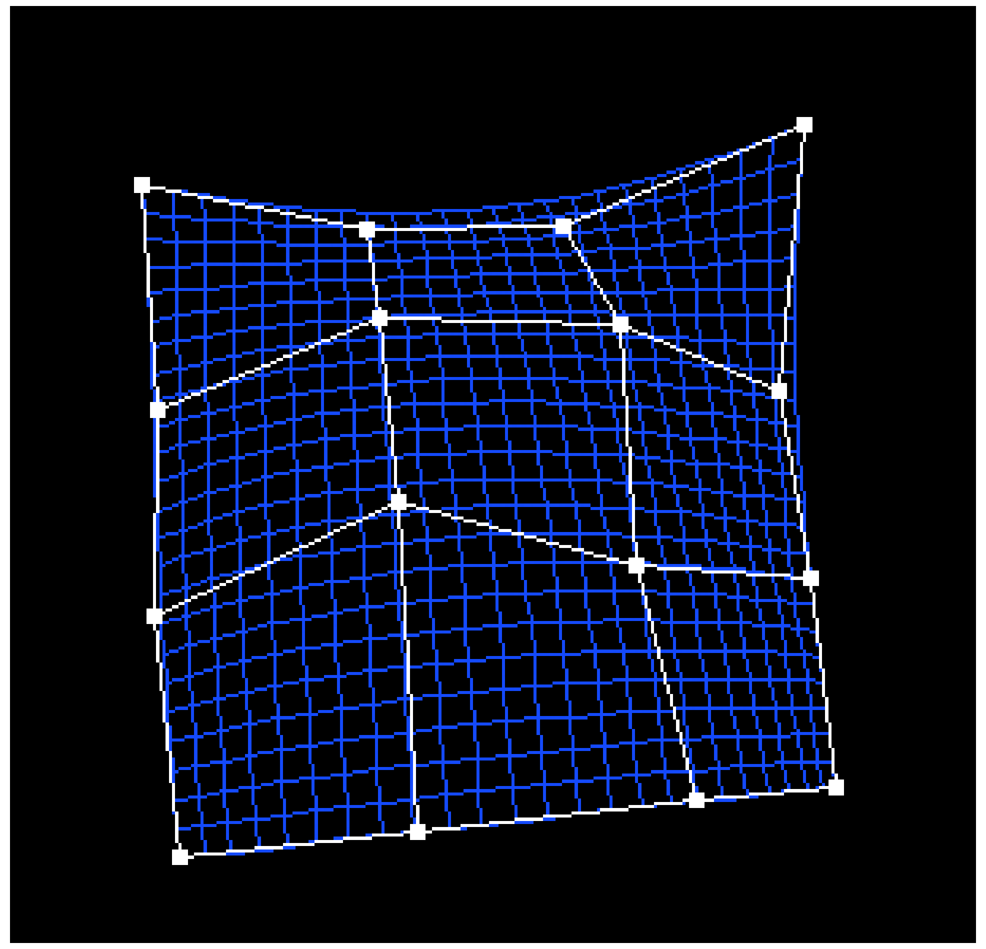
- Traditional pipeline for modeling shapes from scratch



User defines a layout of surface patches and control points

Computer-Aided Geometric Design

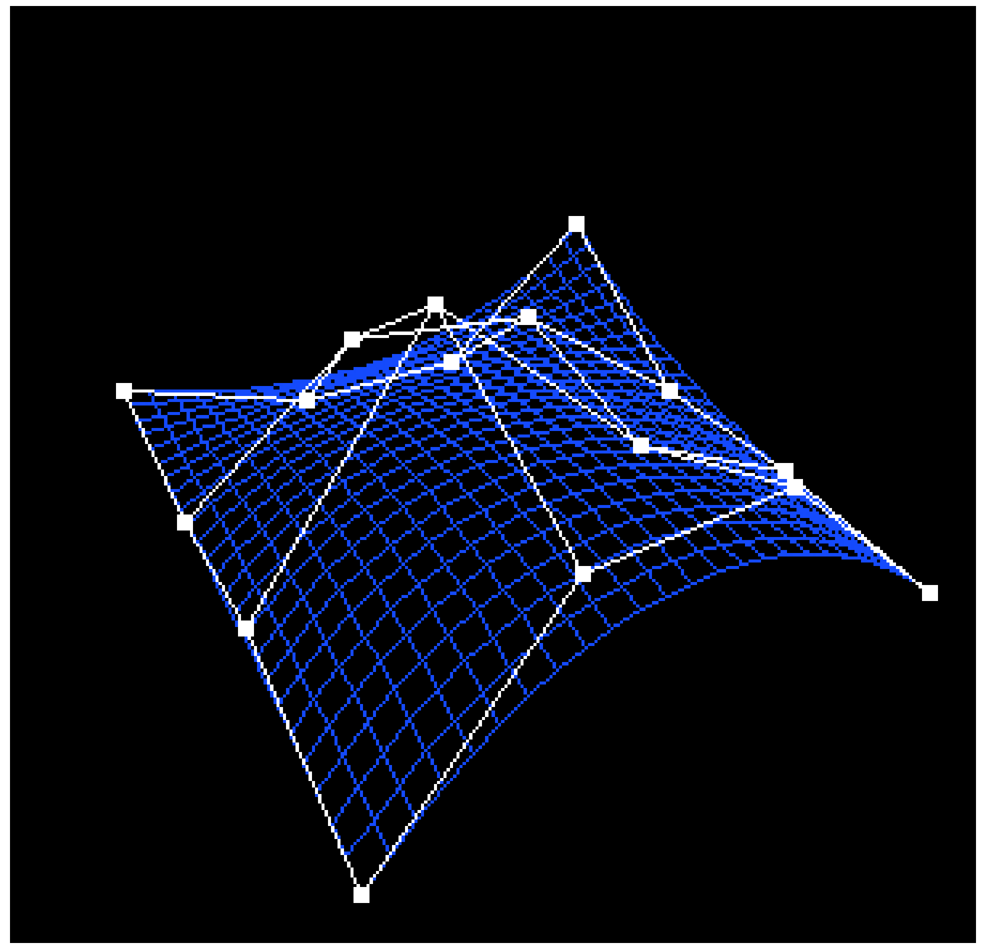
- Traditional pipeline for modeling shapes from scratch



User defines a layout of surface patches and control points

Computer-Aided Geometric Design

- Traditional pipeline for modeling shapes from scratch

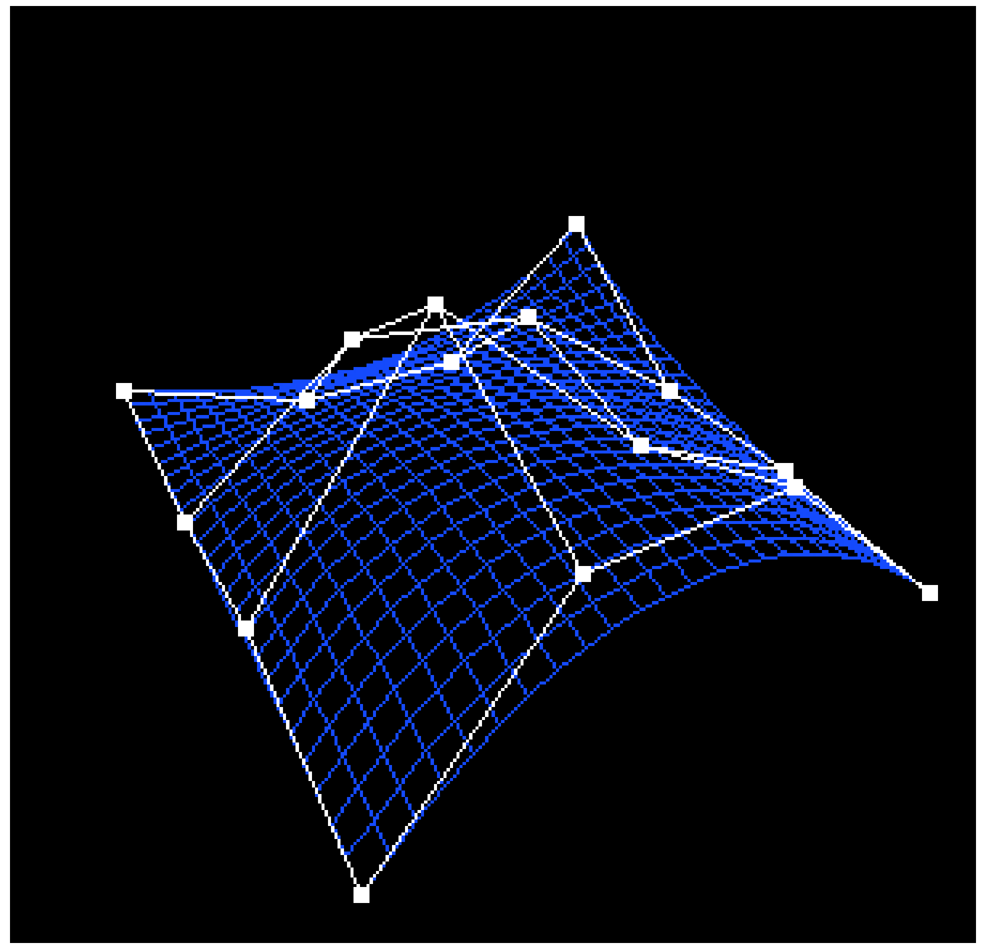


User defines a layout of surface patches and control points

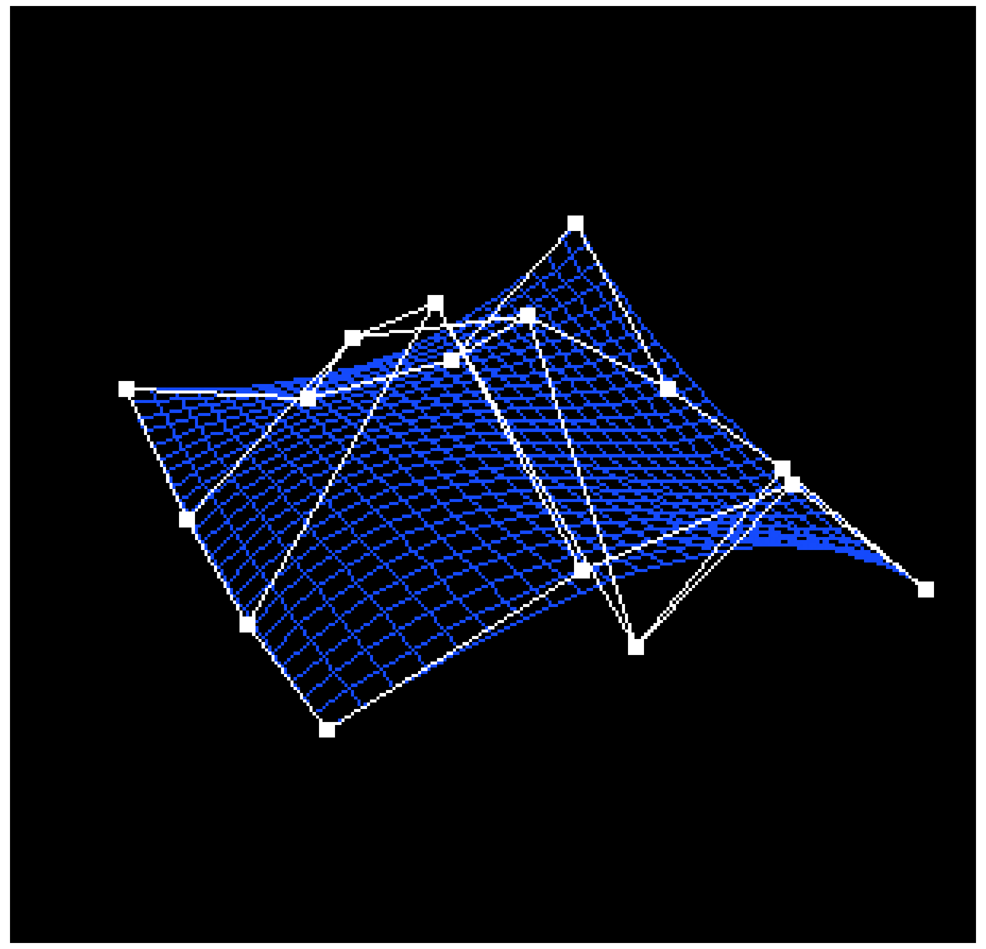
Editing is performed by moving control points and/or prescribing tangents

Computer-Aided Geometric Design

- Traditional pipeline for modeling shapes from scratch



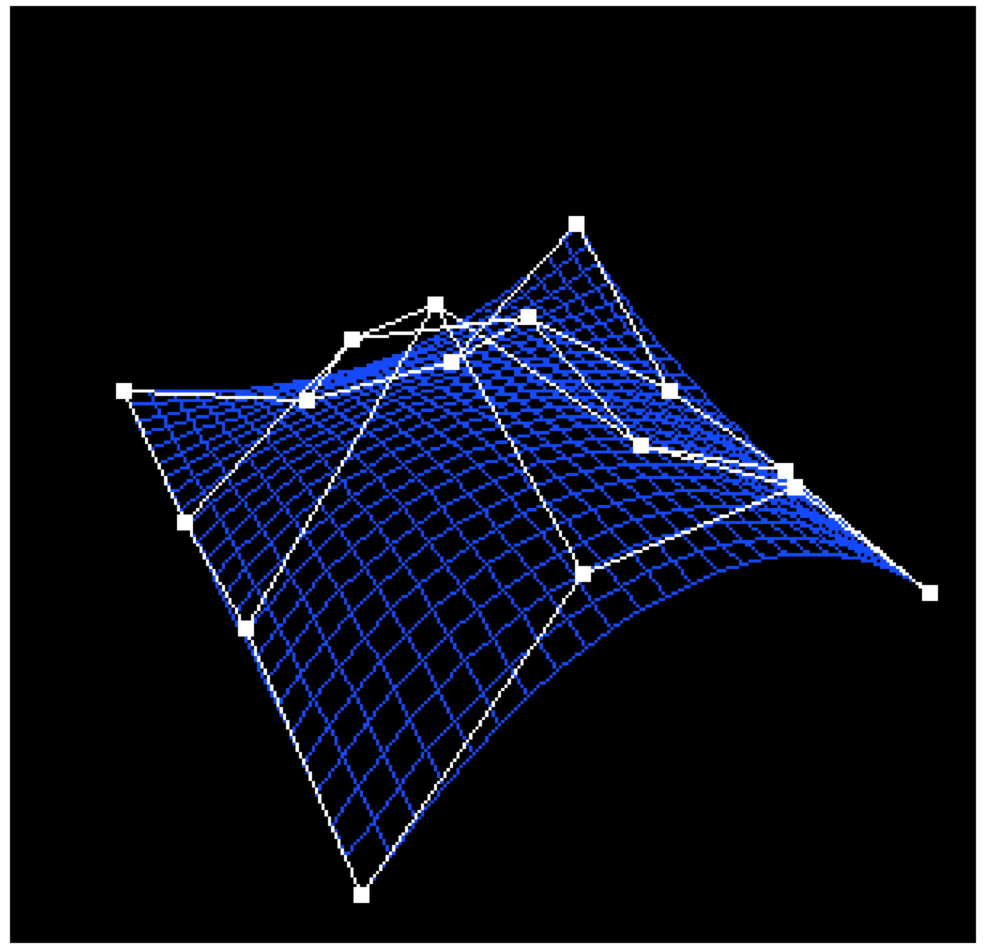
User defines a layout of surface patches and control points



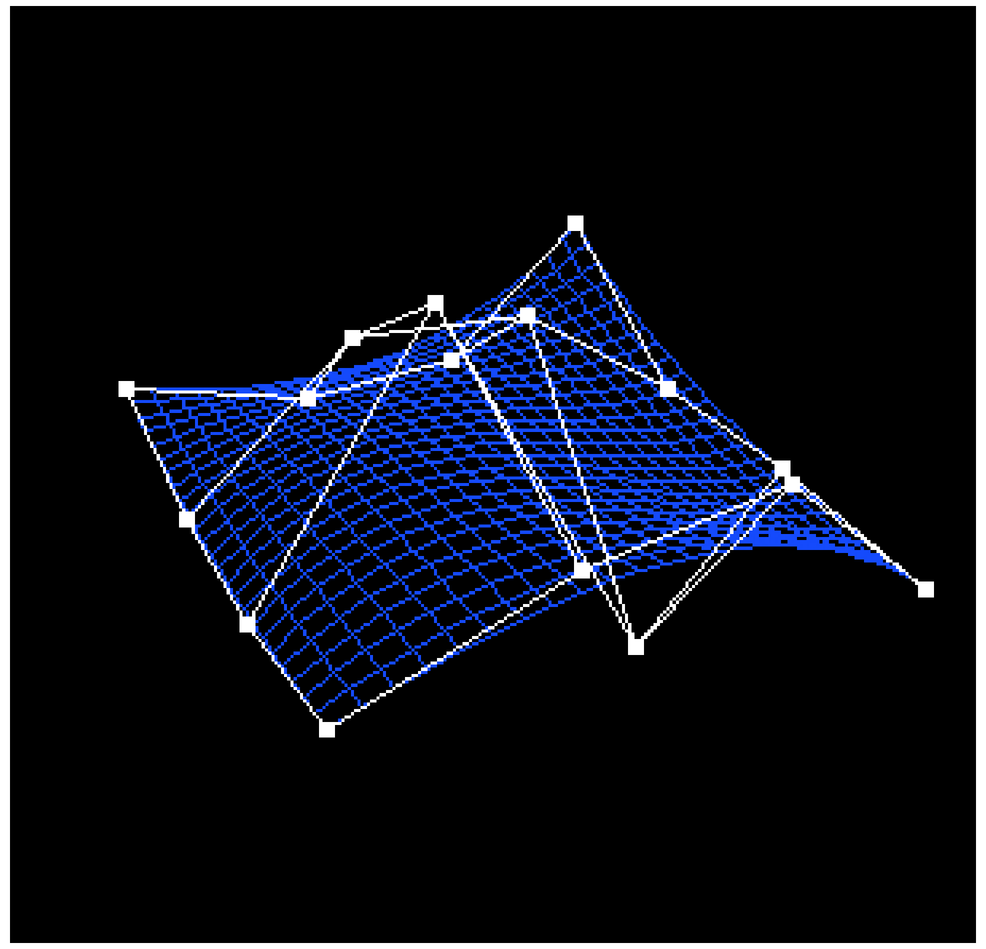
Editing is performed by moving control points and/or prescribing tangents

Computer-Aided Geometric Design

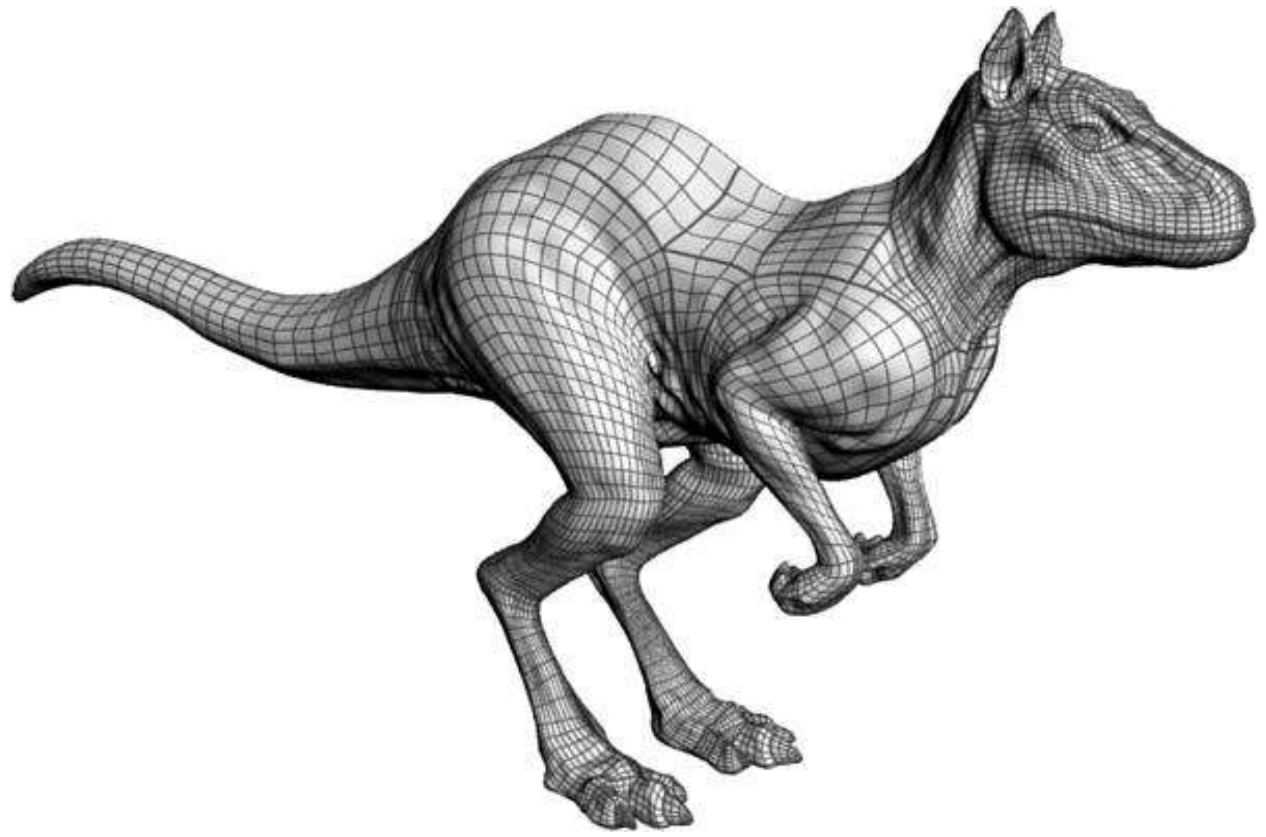
- Traditional pipeline for modeling shapes from scratch



User defines a layout of surface patches and control points

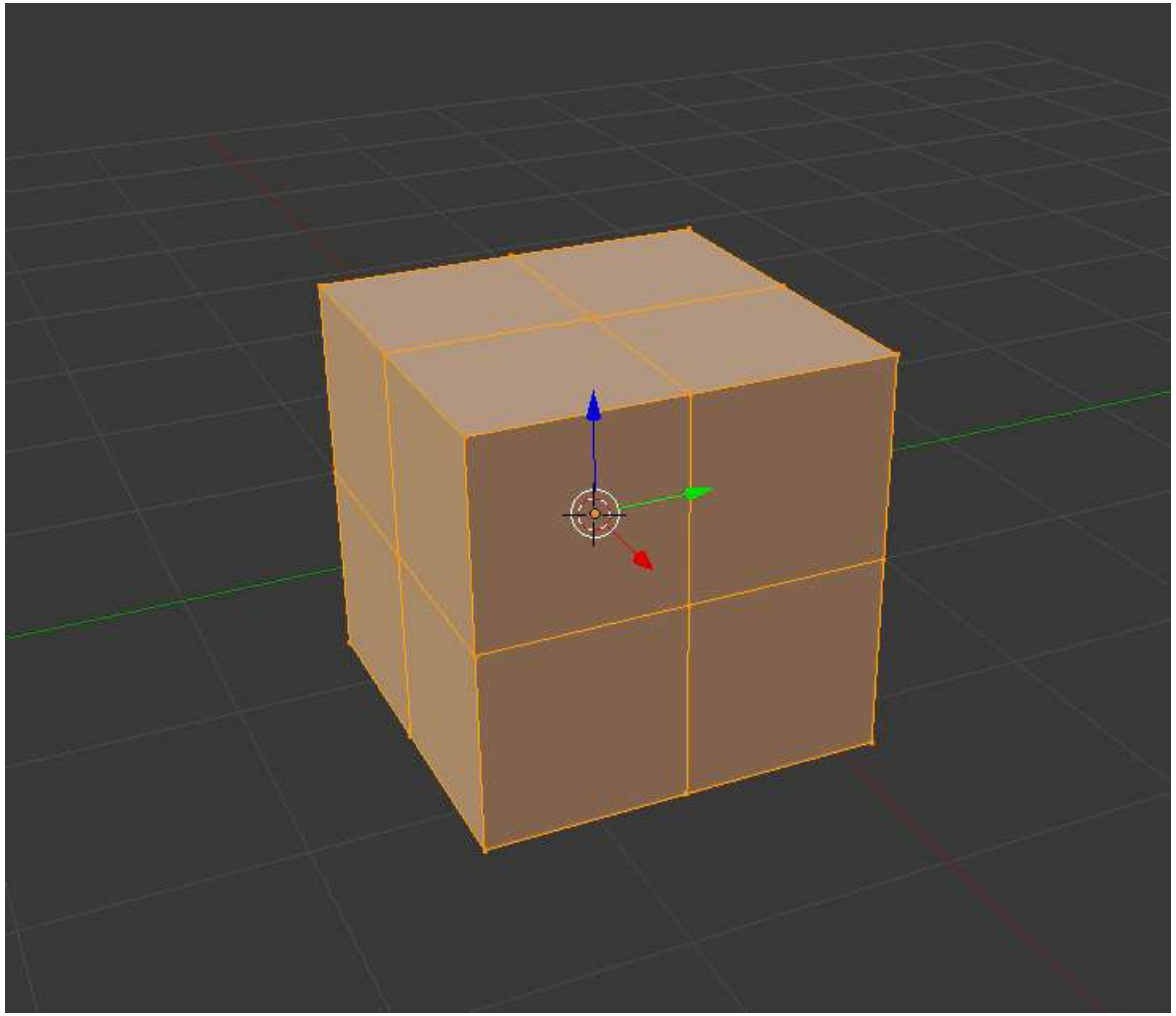


Editing is performed by moving control points and/or prescribing tangents



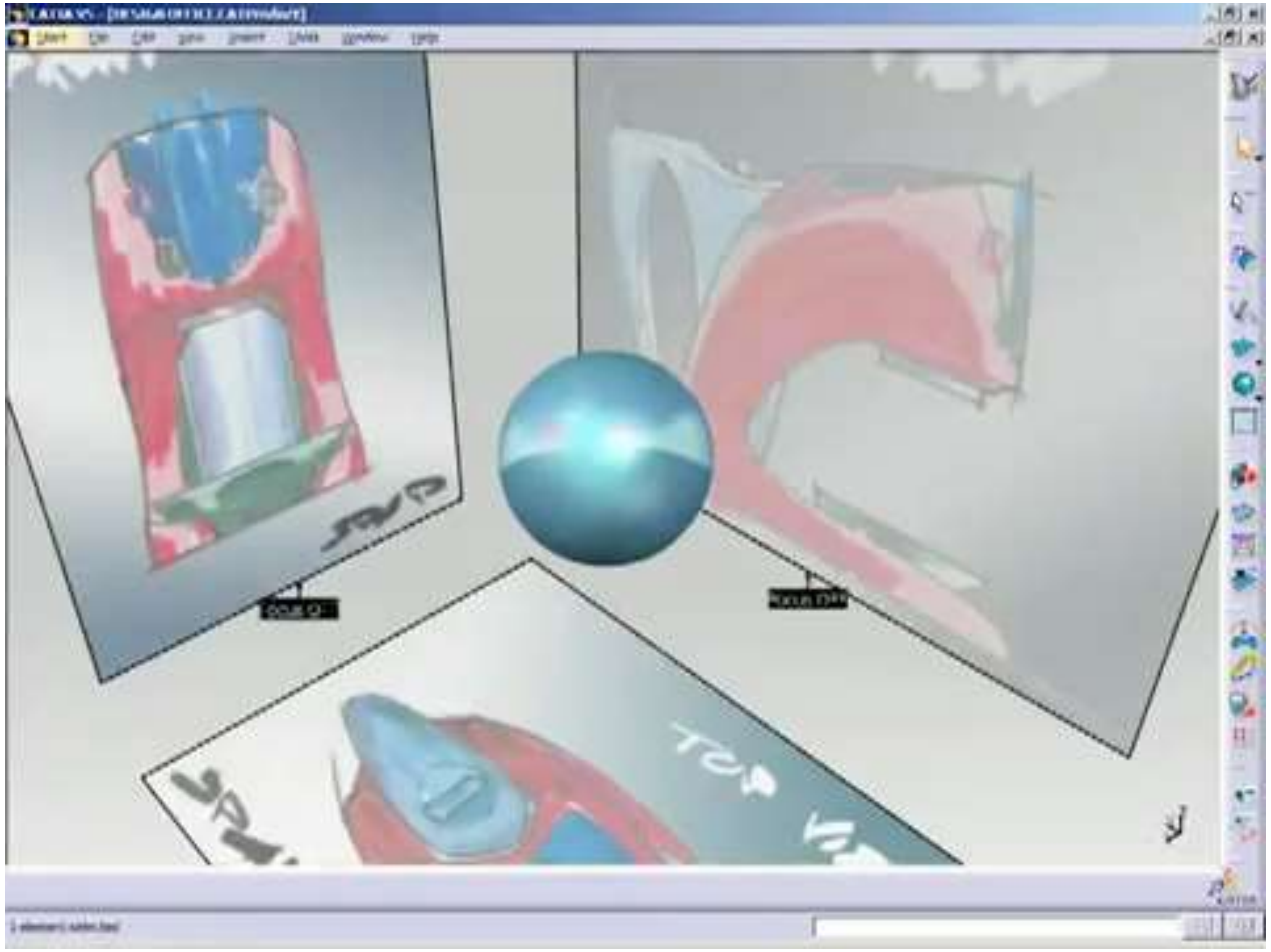
Patch-based construction of a surface

Blender Demo



Computer-Aided Geometric Design

- High-quality surfaces
- Constrained modeling
- Requires a specific idea of the object first
  - Not easy to experiment and explore alternatives
- Requires training, skill and tedious work

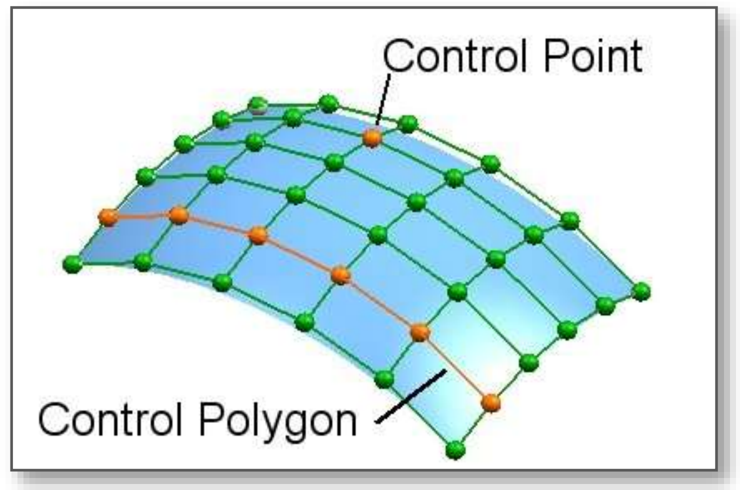


CATIA, Dassault Systemes

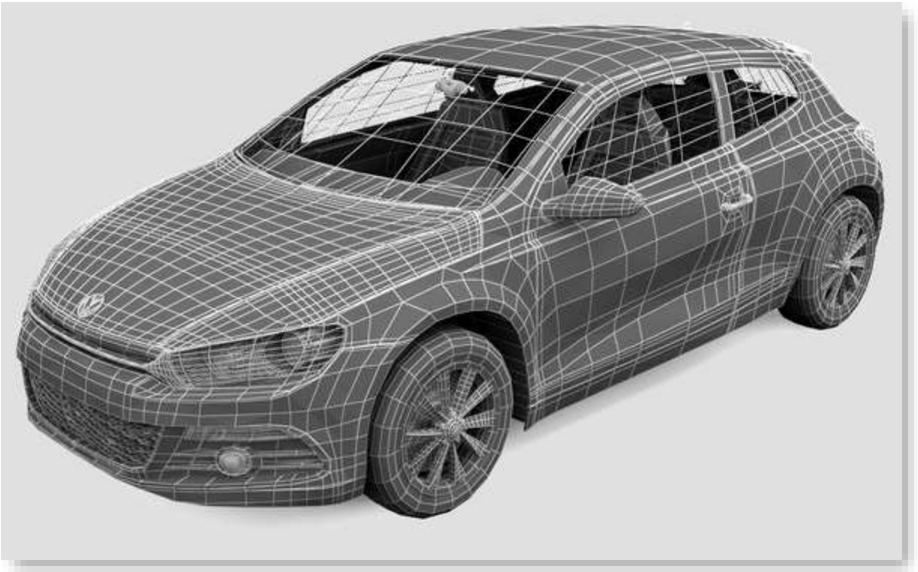
http://youtu.be/gTC5zMktMr0

Traditional CAD vs Freeform Mesh Modeling

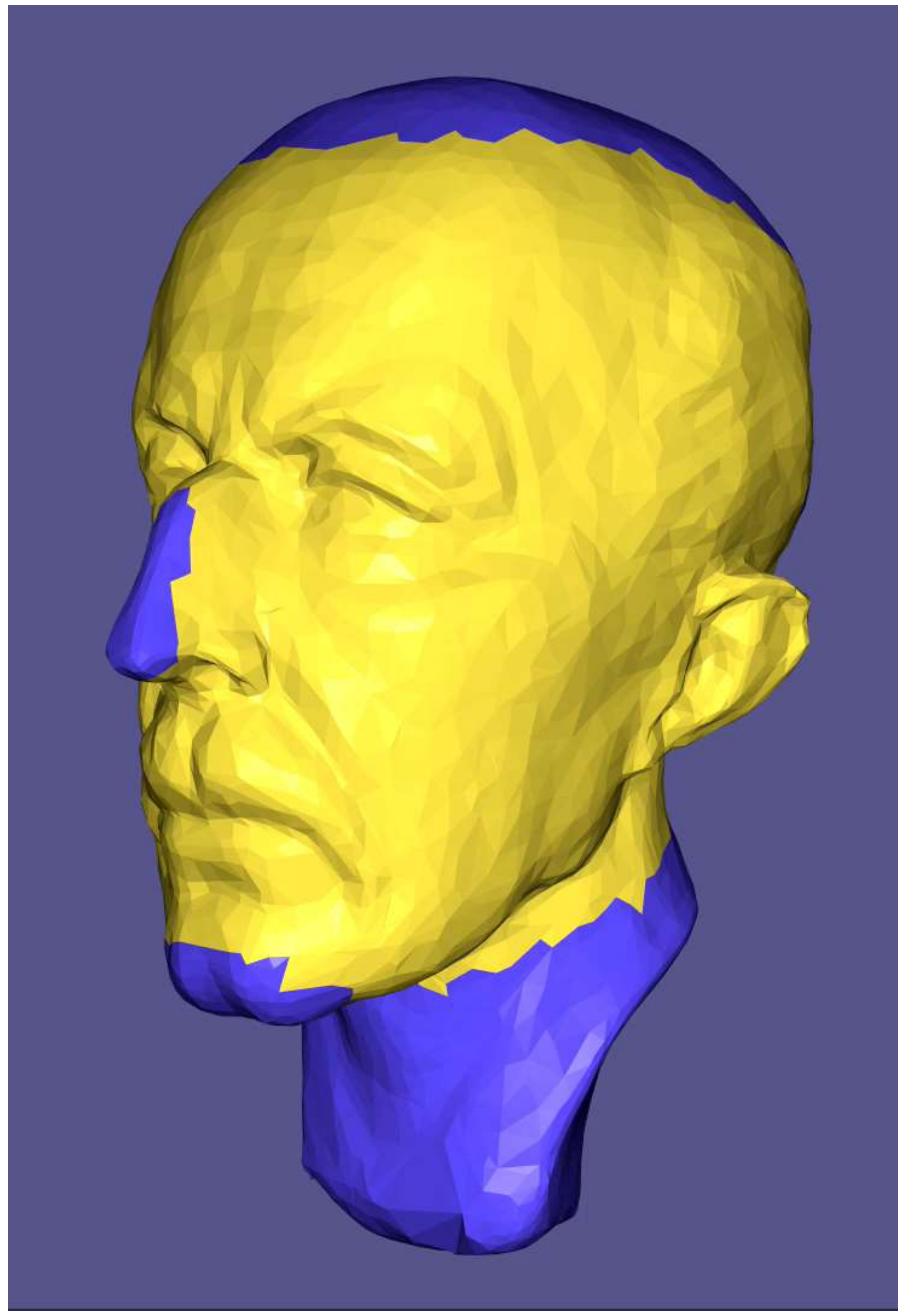
Traditional CAD



$$\mathbf{x}(u,v) = \sum_{i,j} \mathbf{p}_{i,j} B_i(u) B_j(v)$$



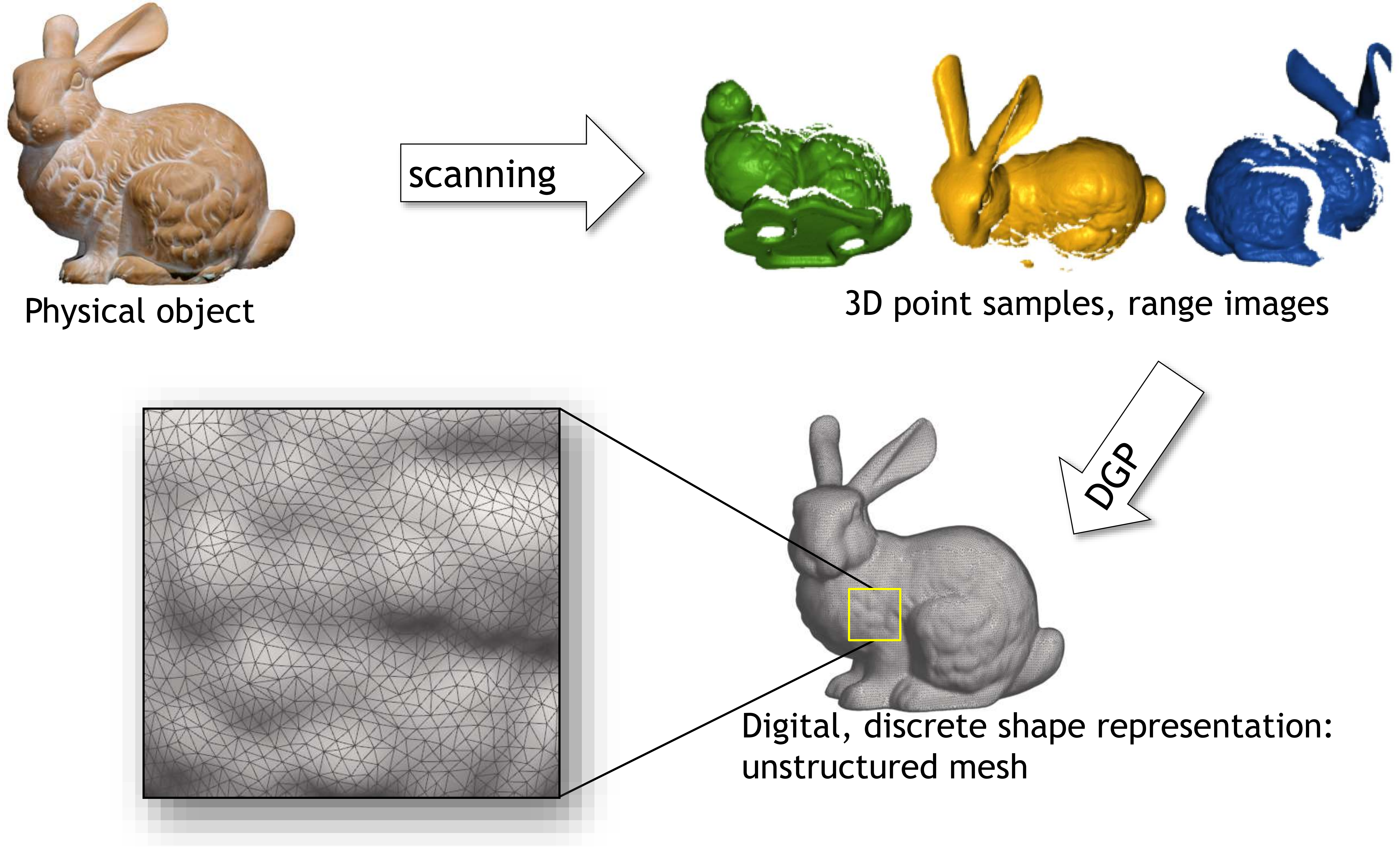
Freeform mesh modeling



$$\min_{\mathbf{x}} \int_{\mathcal{S}} E(\mathbf{x}) \quad s.t. \; \mathbf{x}|_{\mathcal{C}} = \mathbf{x}_{\text{fixed}}$$

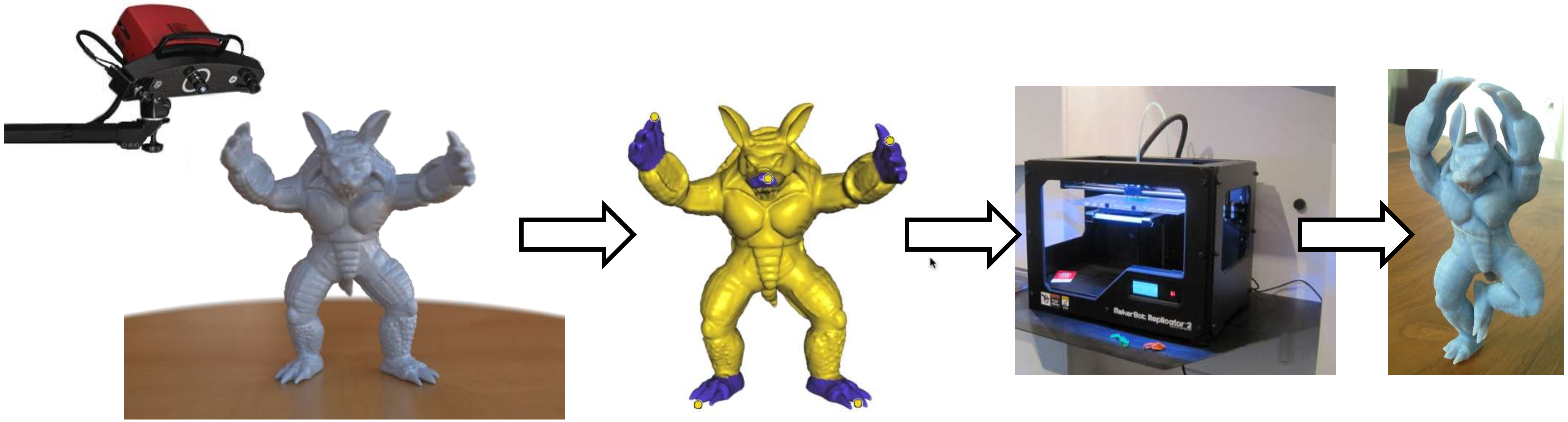
User has more freedom!
Select and manipulate arbitrary regions.

Scanning-based Geometry Acquisition Pipeline



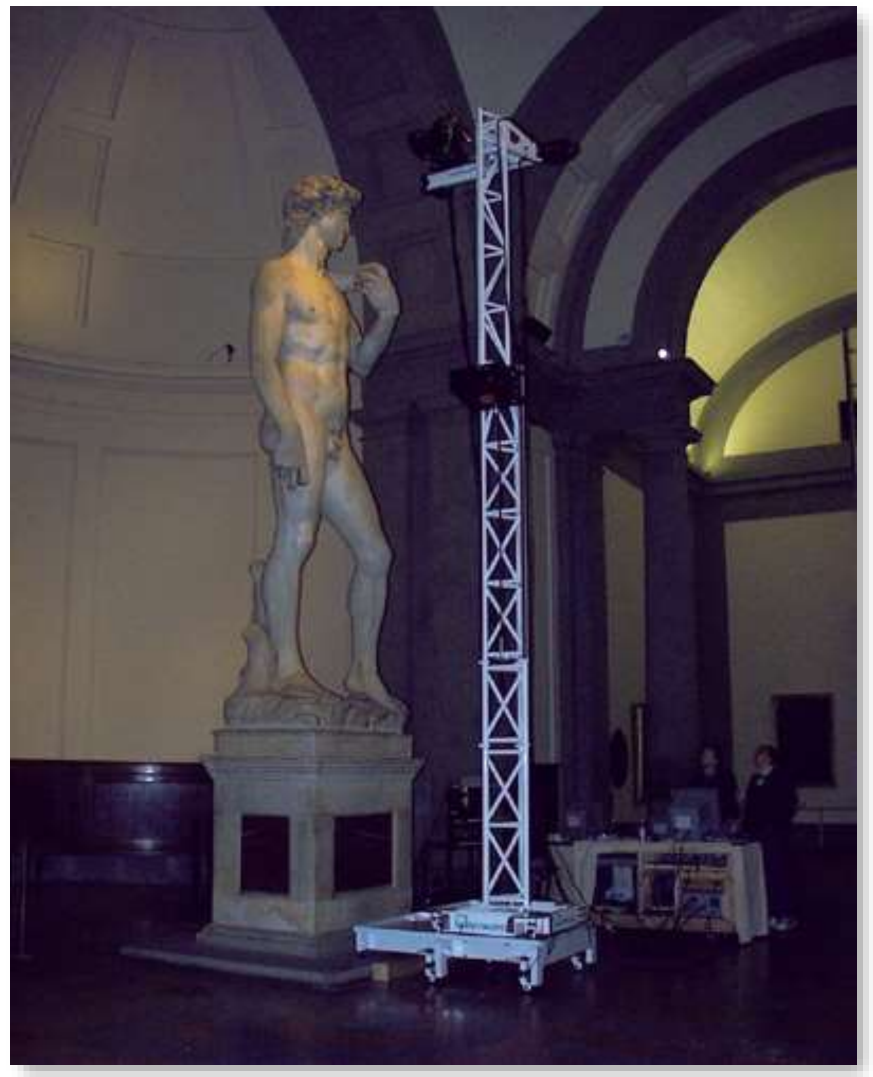
Fabrication

- Modern scanning and 3D printing technologies allow replication and much more



Digital Geometry Processing (DGP)

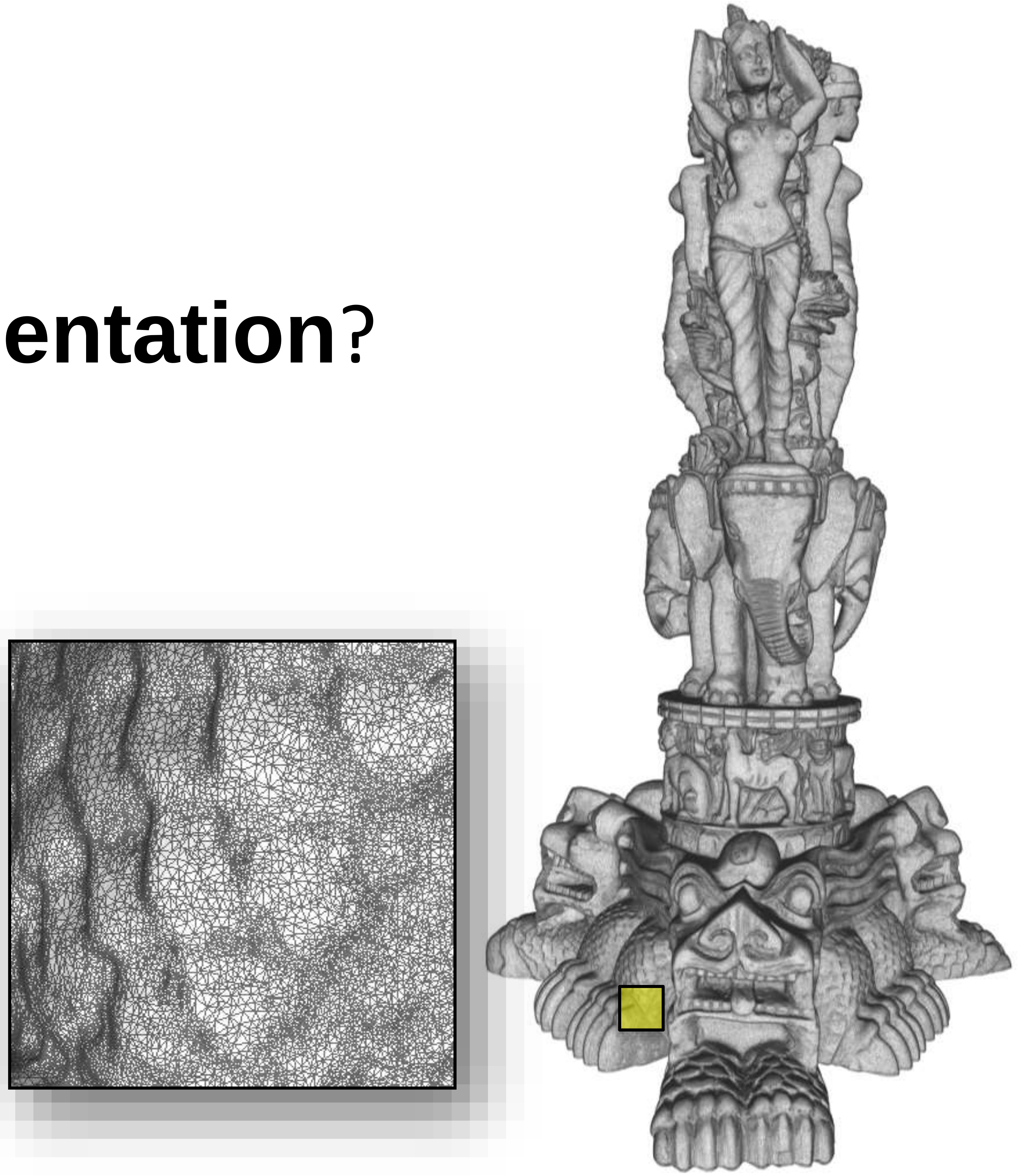
- Processing of discrete (polygonal mesh) models
- Why discrete?
  - Simplicity – ease of description
  - Efficiently rendered by graphics hardware
  - Output of most acquisition tools (CT, MRI, LIDAR, Kinect…)
  - Input to most simulation/analysis tools (FE solvers)



The Digital Michelangelo Project

Unstructured Digital Shapes

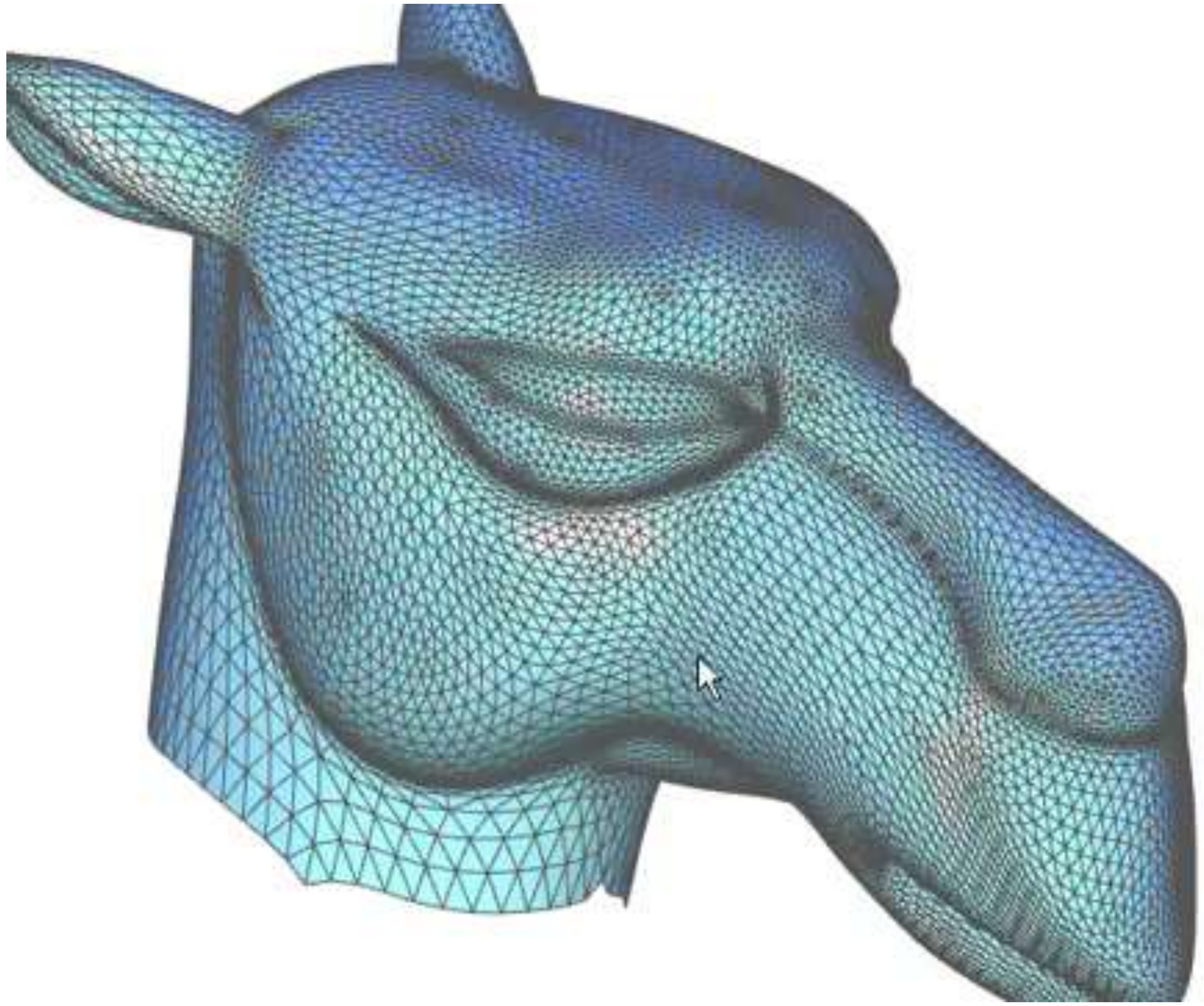
- How to **edit** and **animate**?
- How to convert to a **structured representation**?
- Computational challenge:
  very large amounts
  of data, yet modeling has to
  remain interactive



Thai statue, 10M triangles, Stanford 3D Scanning Repository

Interactive Shape Modeling

- Tools for design, editing and animation of digital shapes
  - Interactive means fast algorithms
  - Intuitive – convenient interface and predictable outcome



http://youtu.be/EMx6yNe23ug

Tools?

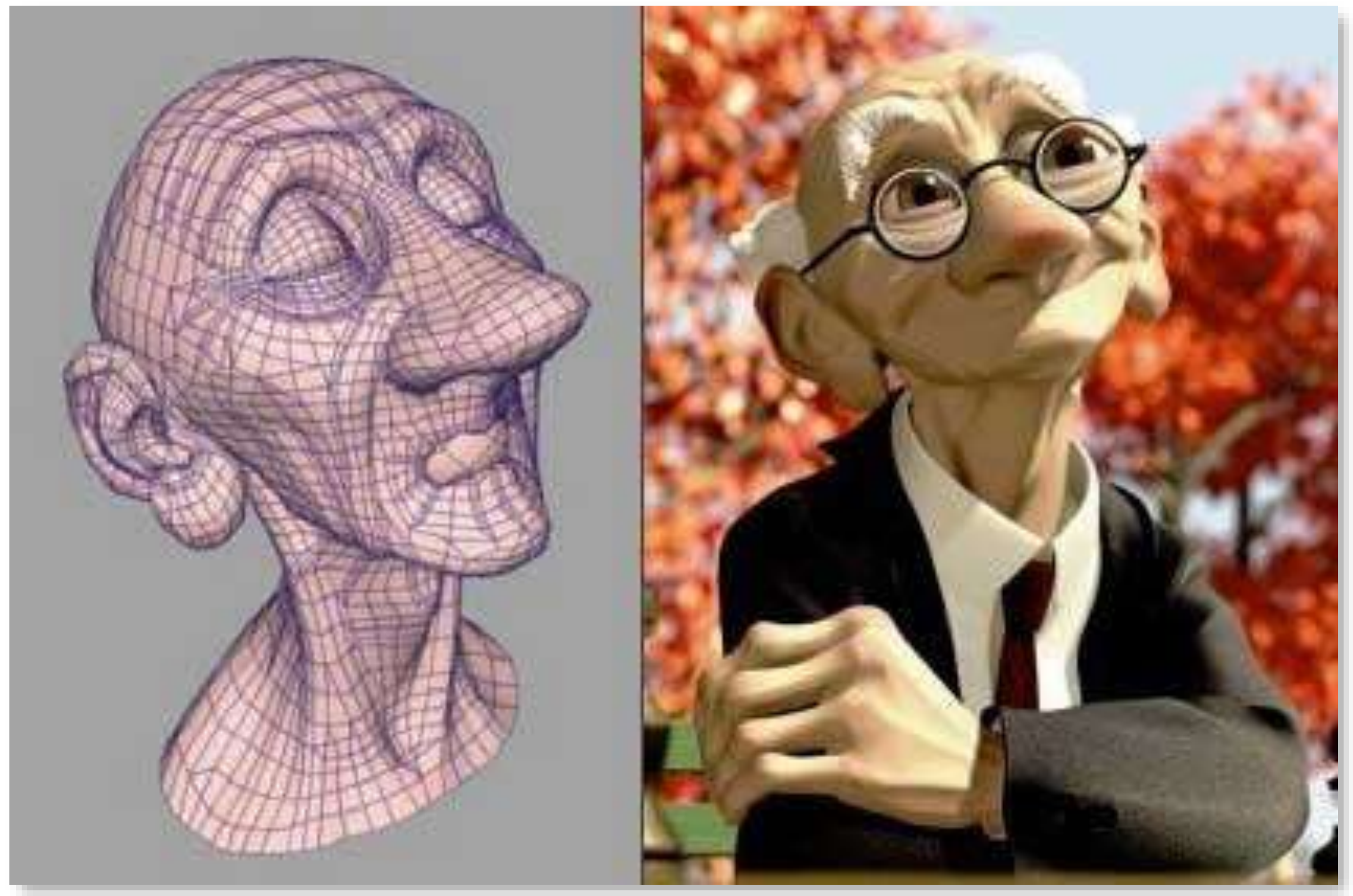
- Use techniques from both CS & Math
  - PDEs
  - Discrete differential geometry
  - Numerical linear algebra
  - Graph theory
  - ...
- ...combined with intuition and creativity ...
- work on real data = write/use code

Break

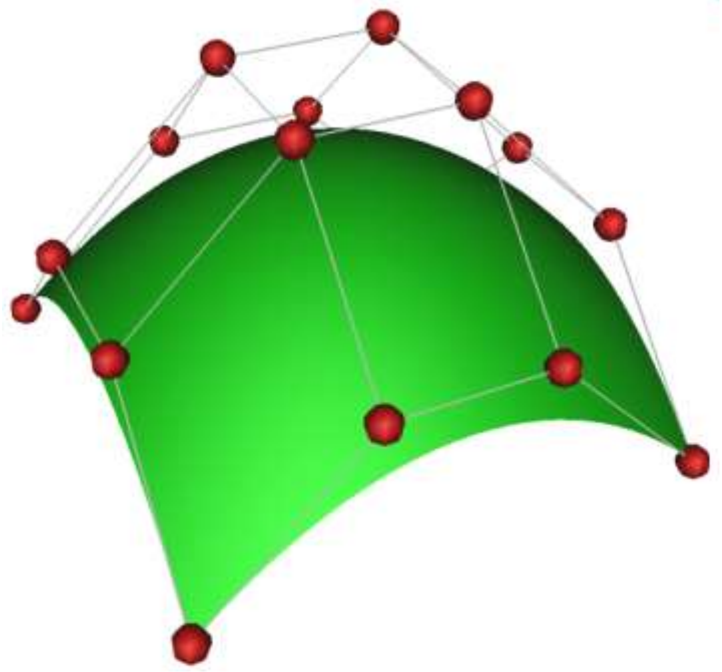
- What do you prefer?
- Let me know after this 10 minutes break!

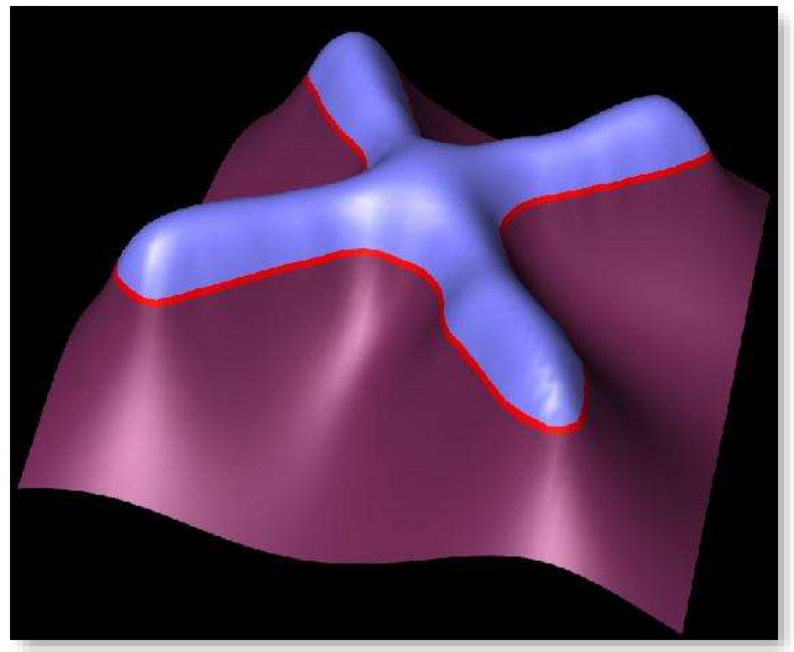
Course Topics

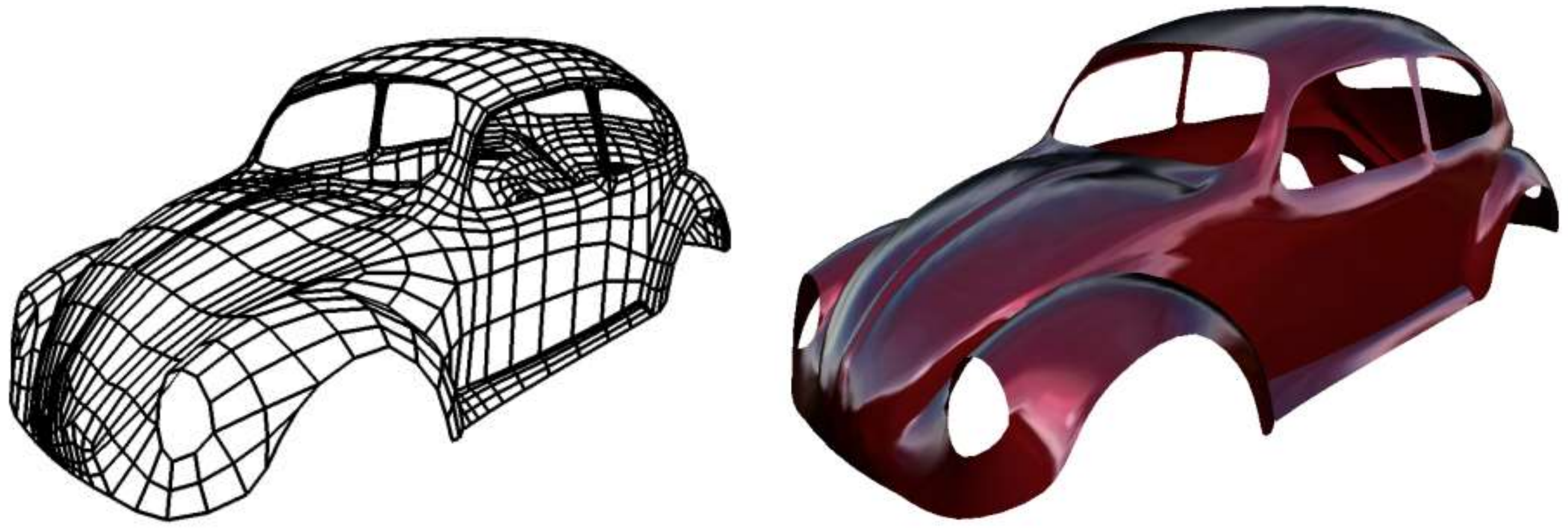
- Overview of shape representations
  - Parametric curves/surfaces
  - Implicits
  - Polygonal meshes



Pixar

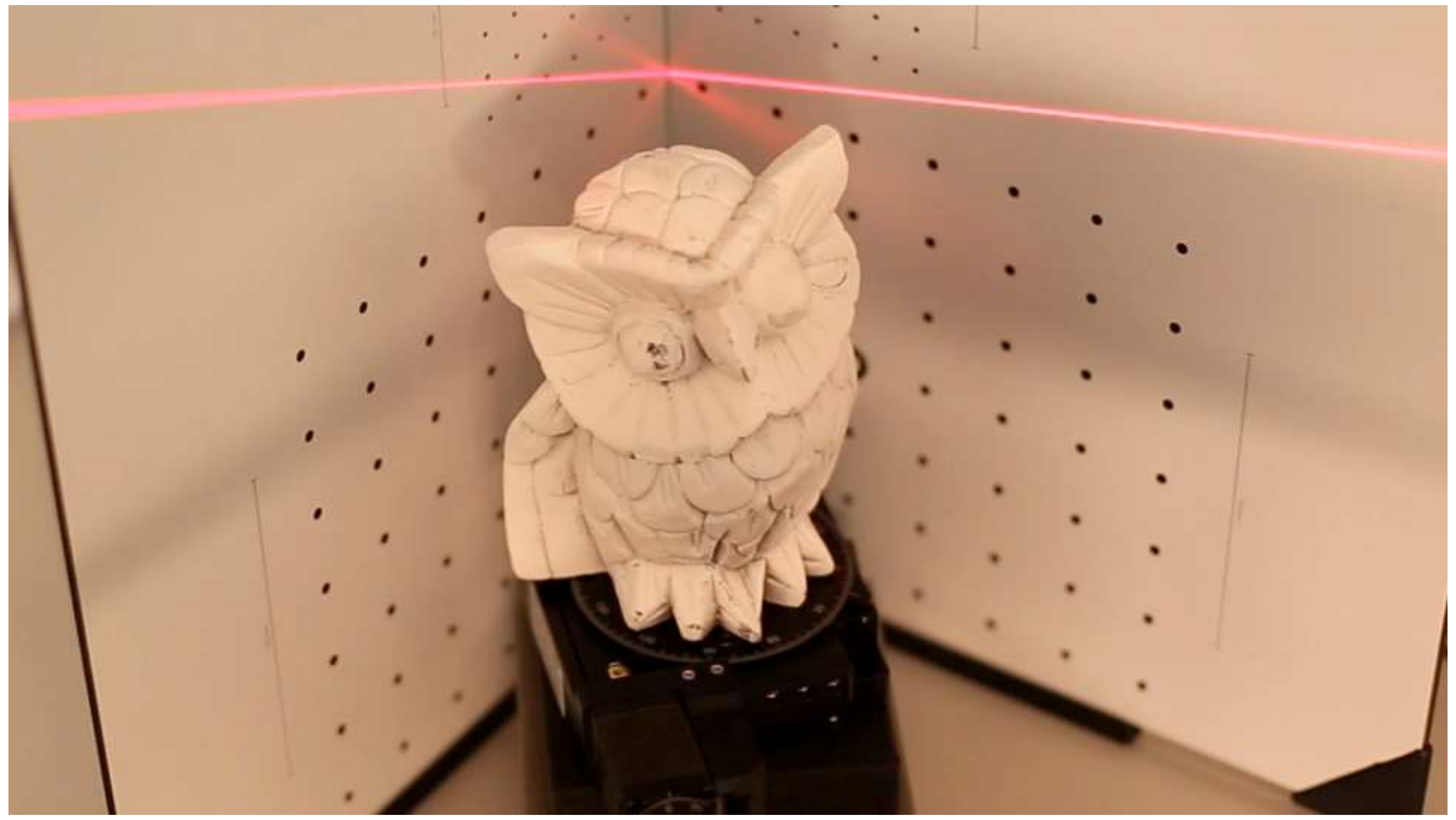


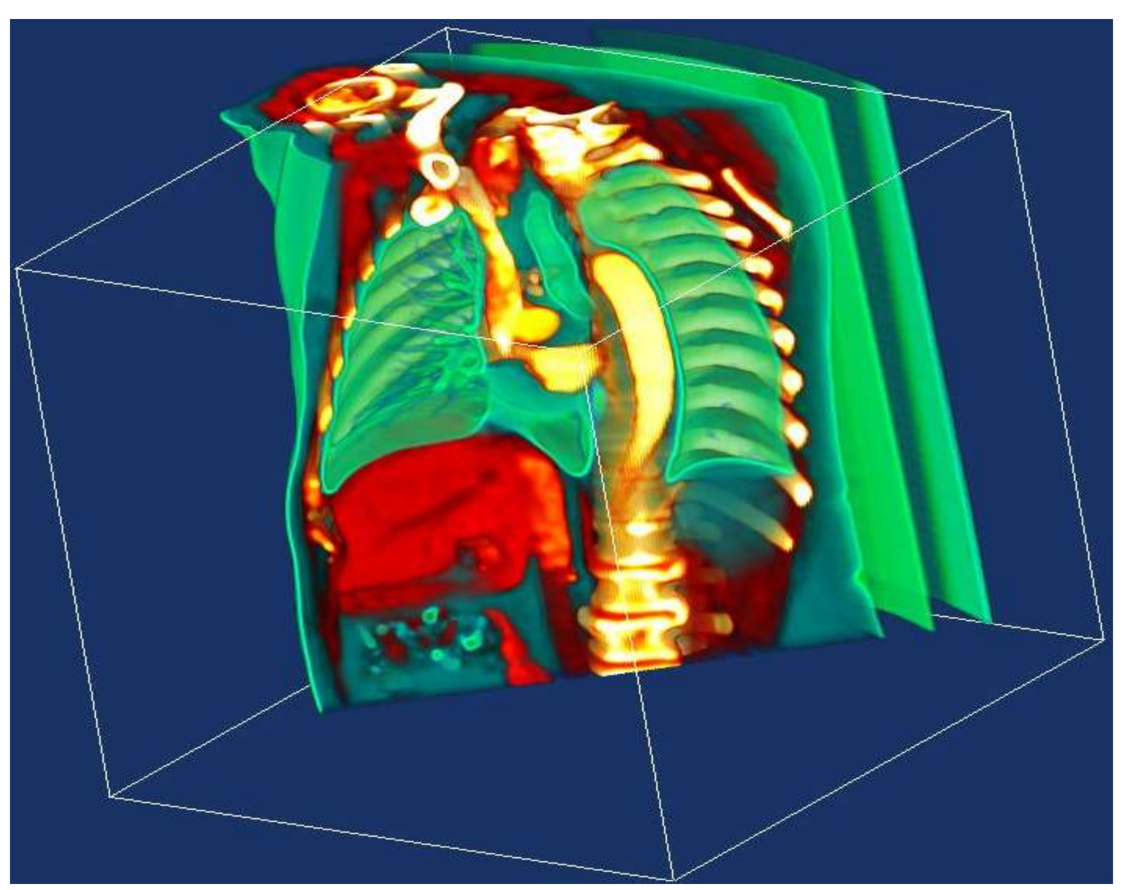


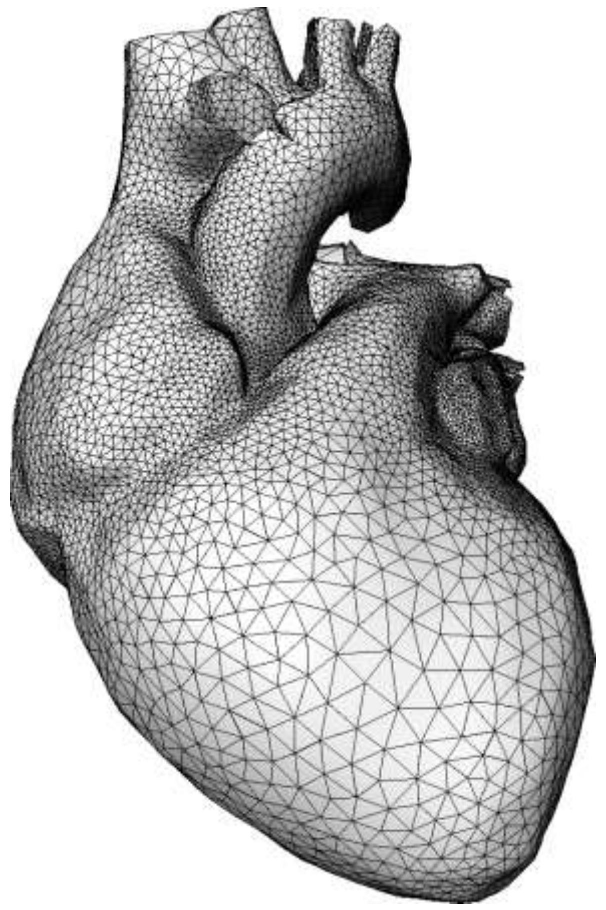


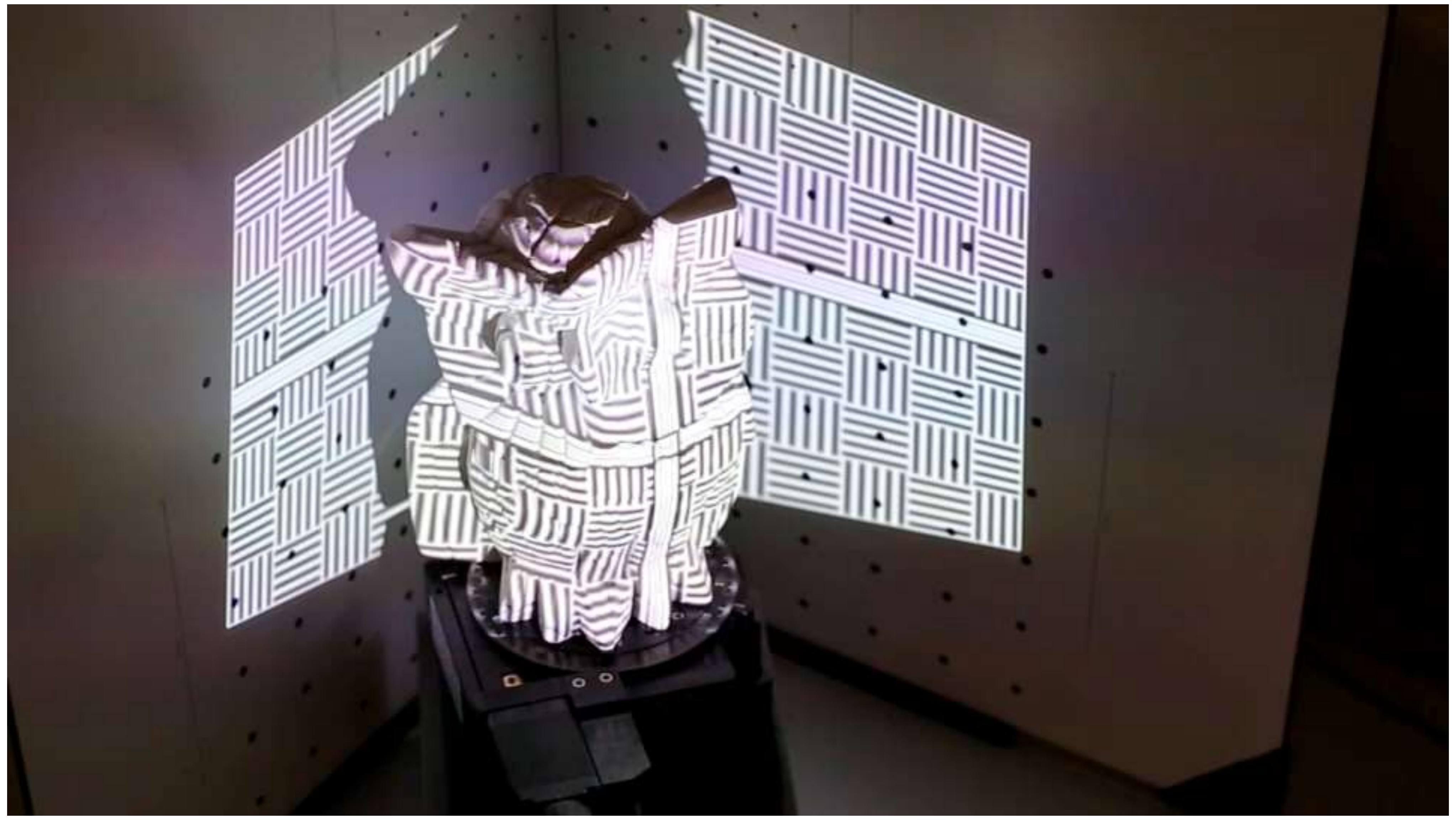
Course Topics

- Shape acquisition
  - Scanning/imaging
  - Reconstruction



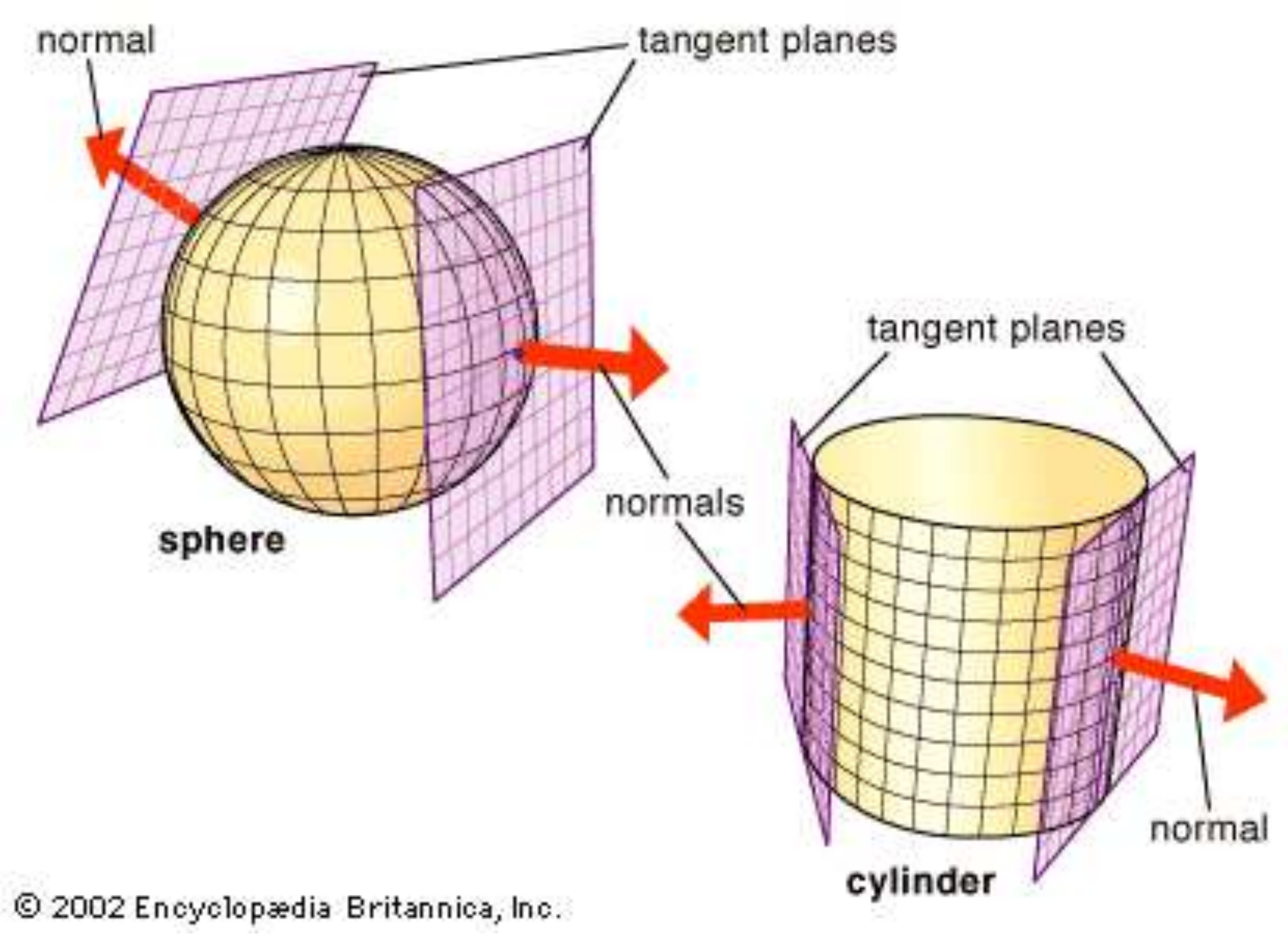


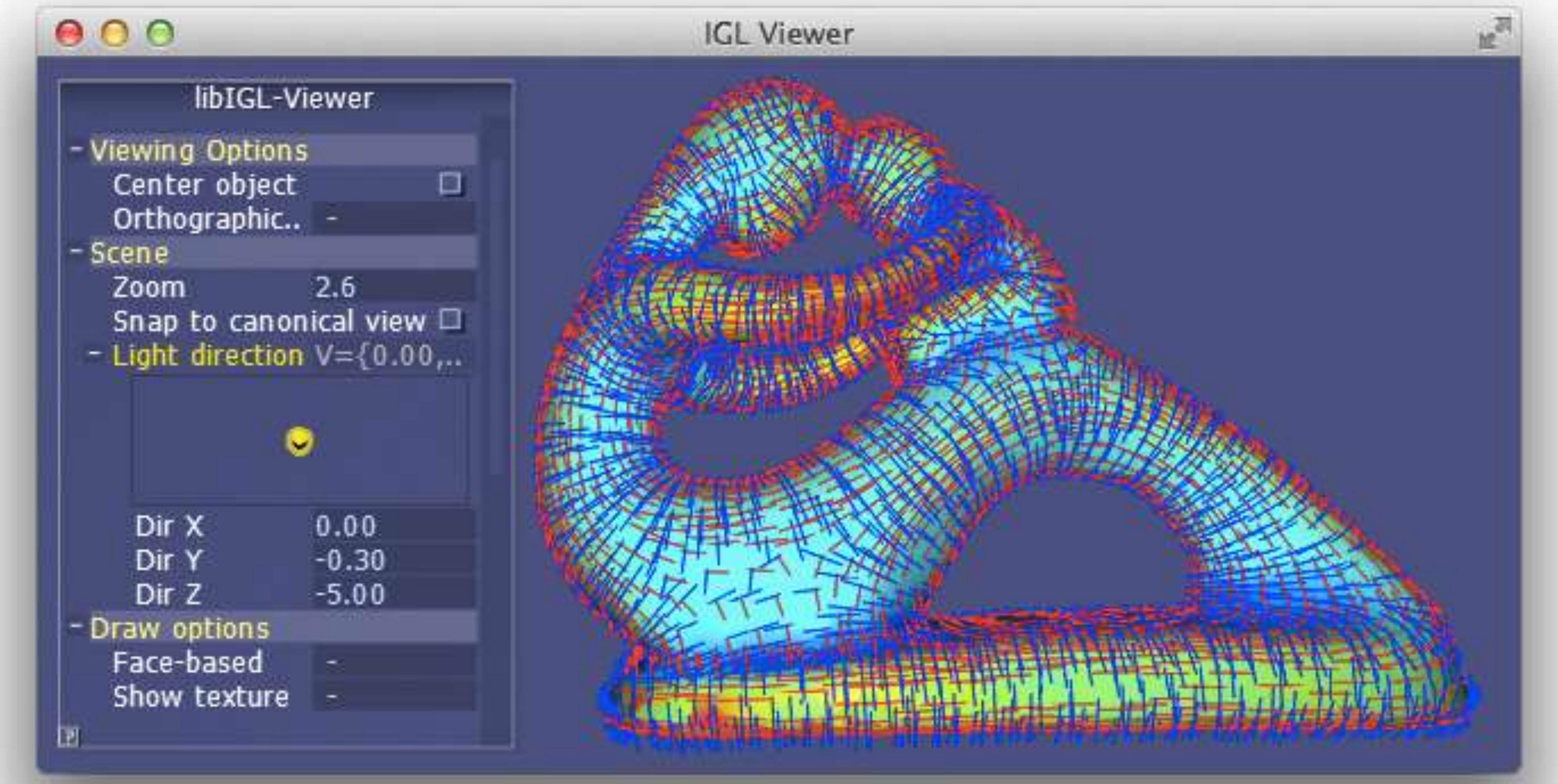




Course Topics

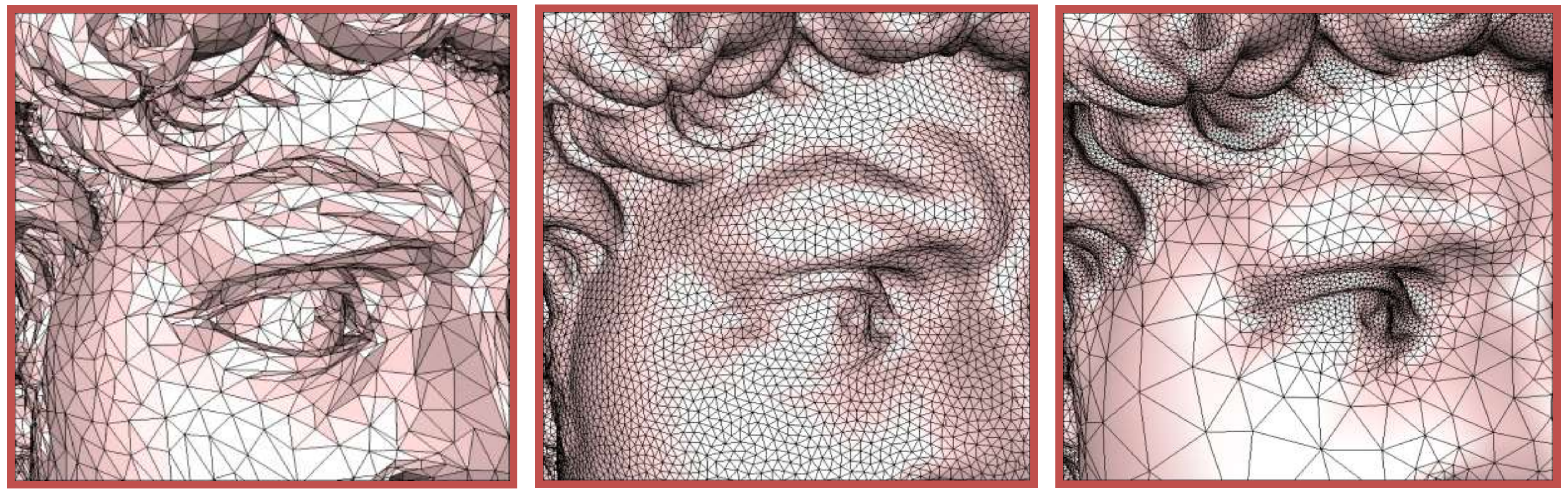
- Differential geometry
  - Continuous and (mostly) discrete
  - Powerful tool to analyze and model shapes

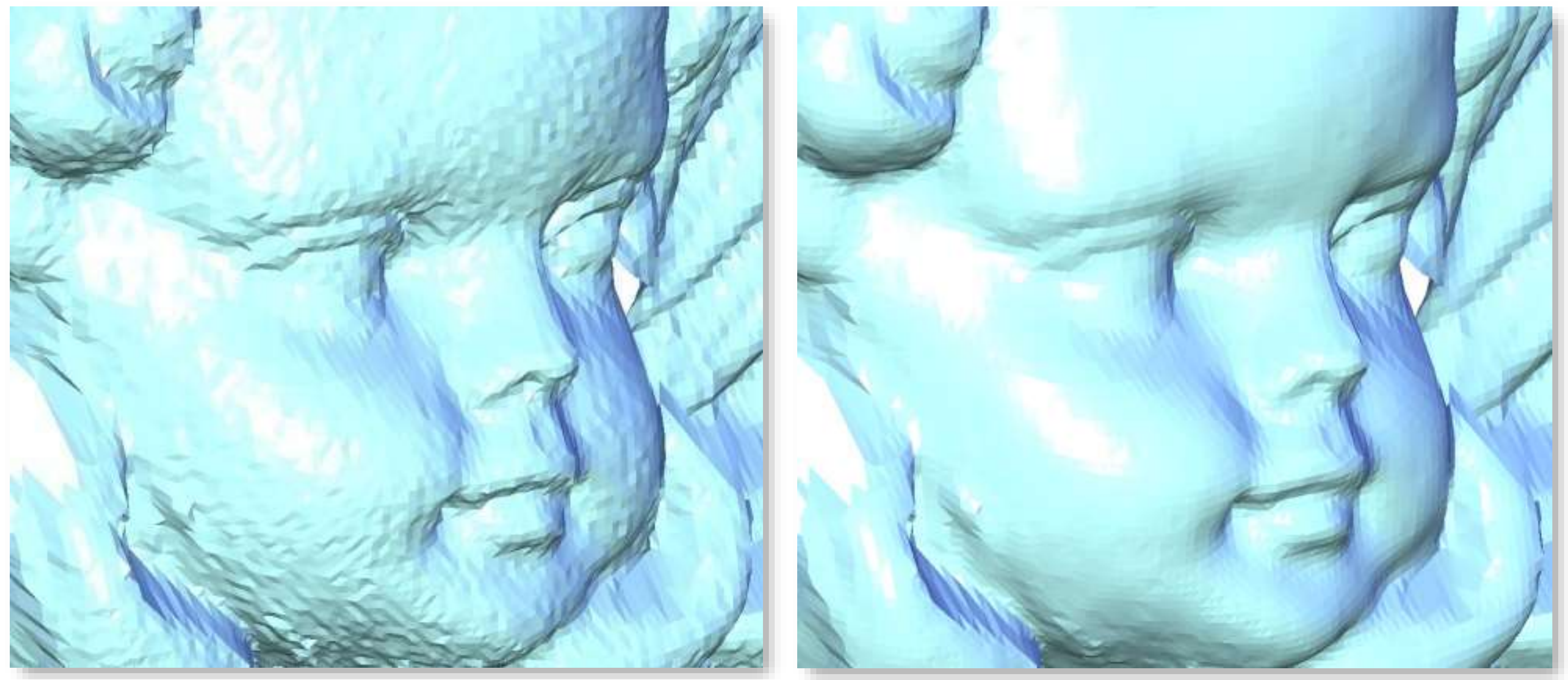




Course Topics

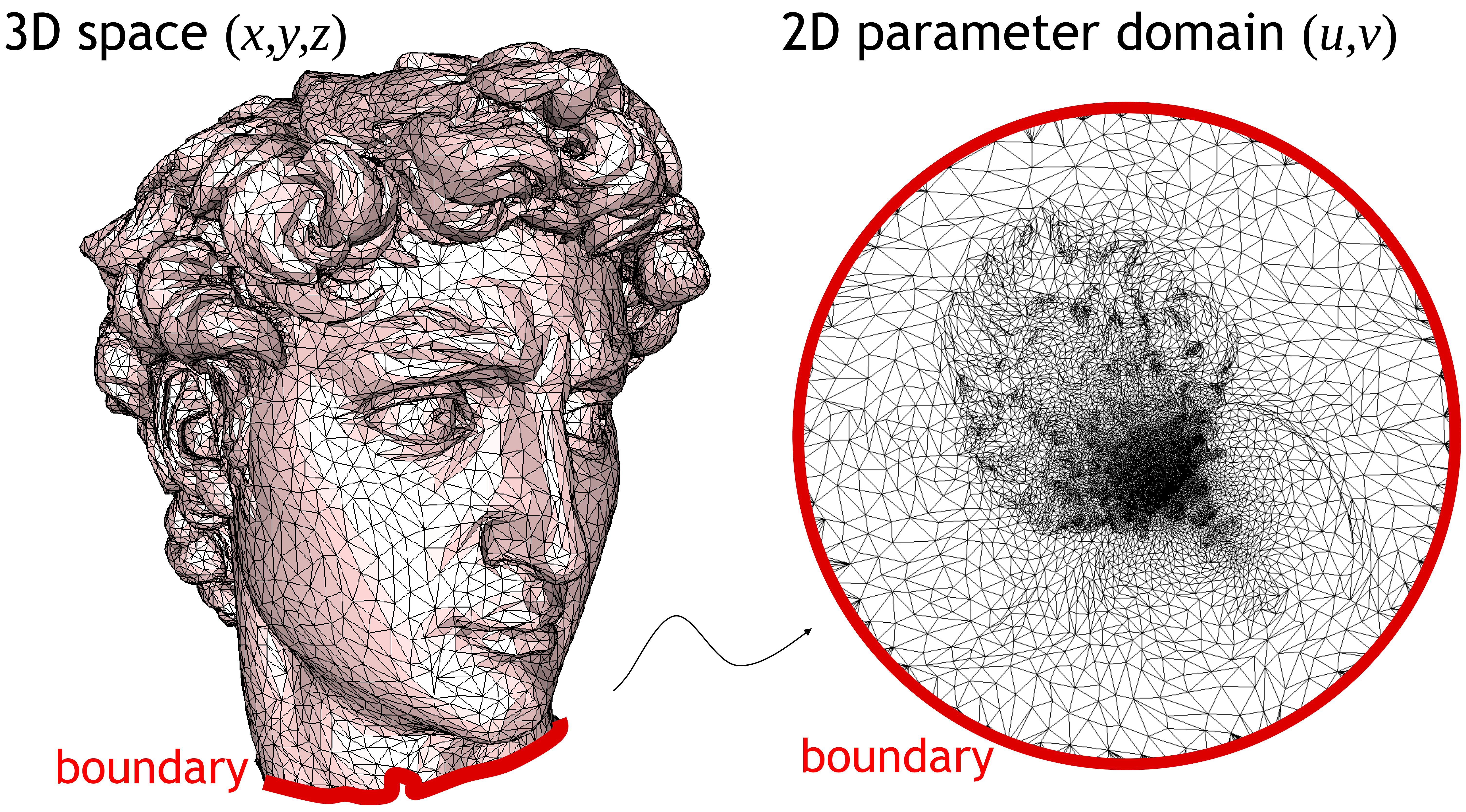
- Digital geometry processing
  - Denoising, smoothing, simplification, remeshing, parameterization, compression





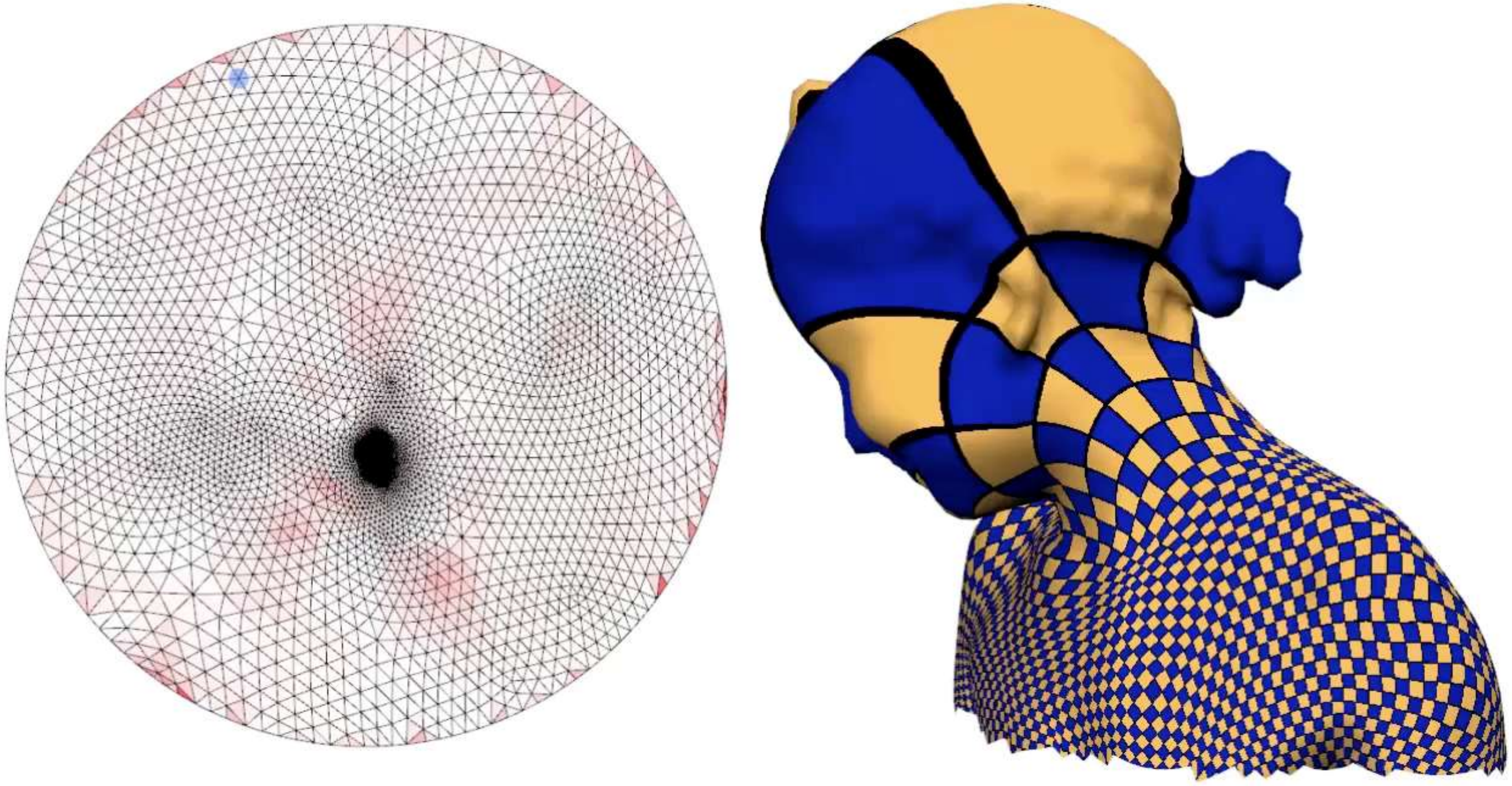
Course Topics

- Parameterization



Course Topics

- Parameterization



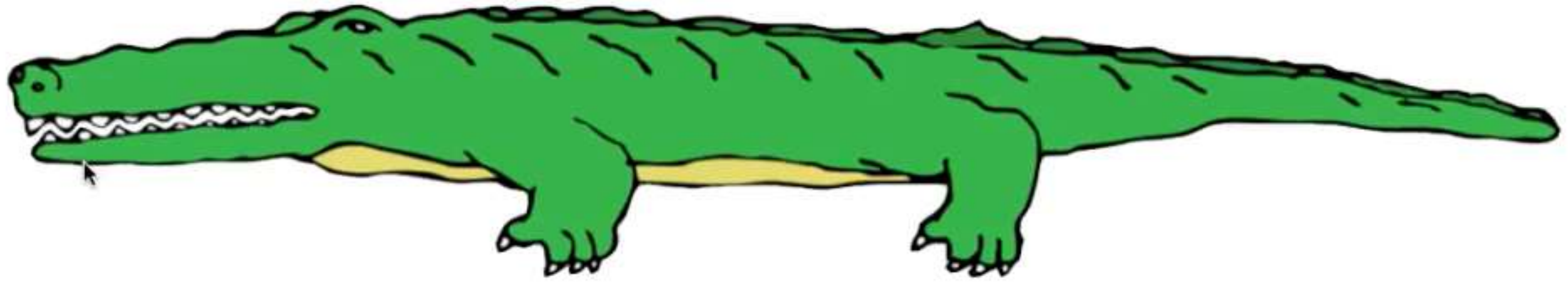
Course Topics

- Shape creation and editing

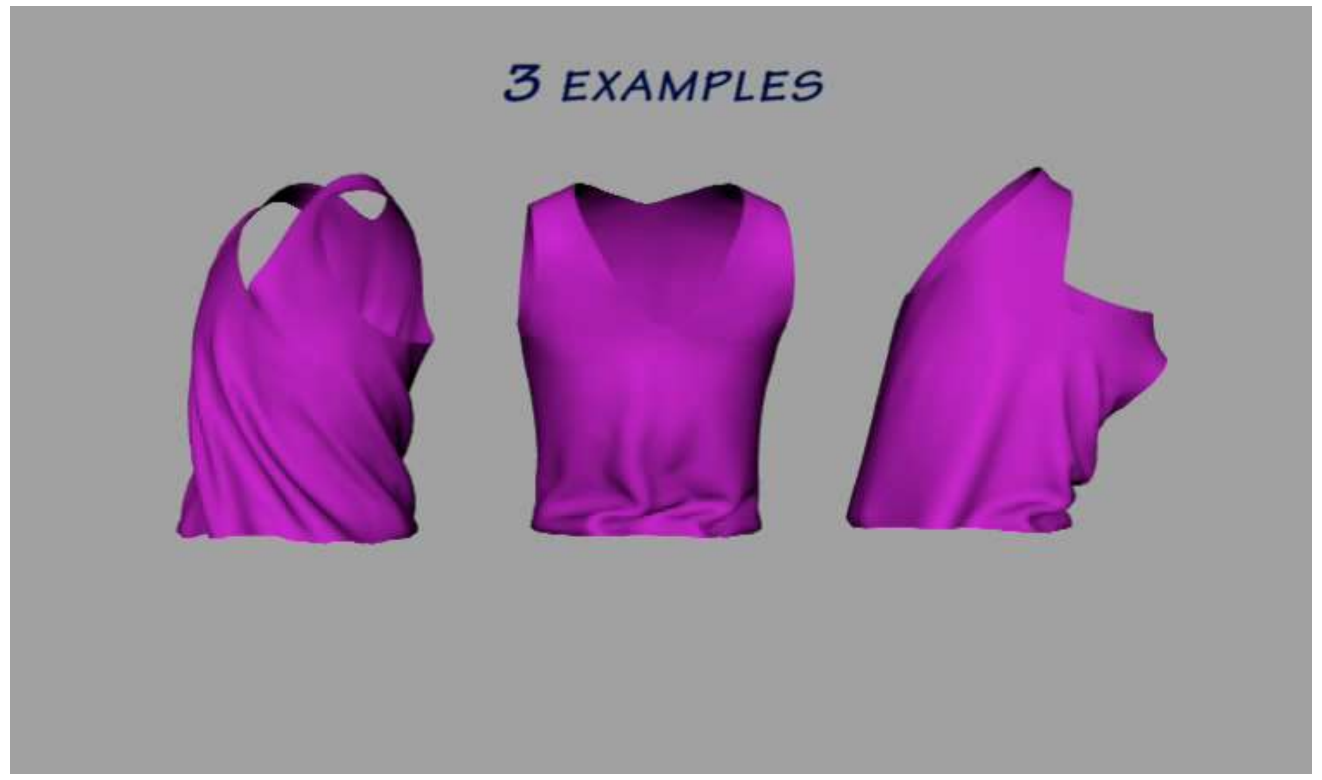
http://youtu.be/W0XGkS7zebo

Course Topics

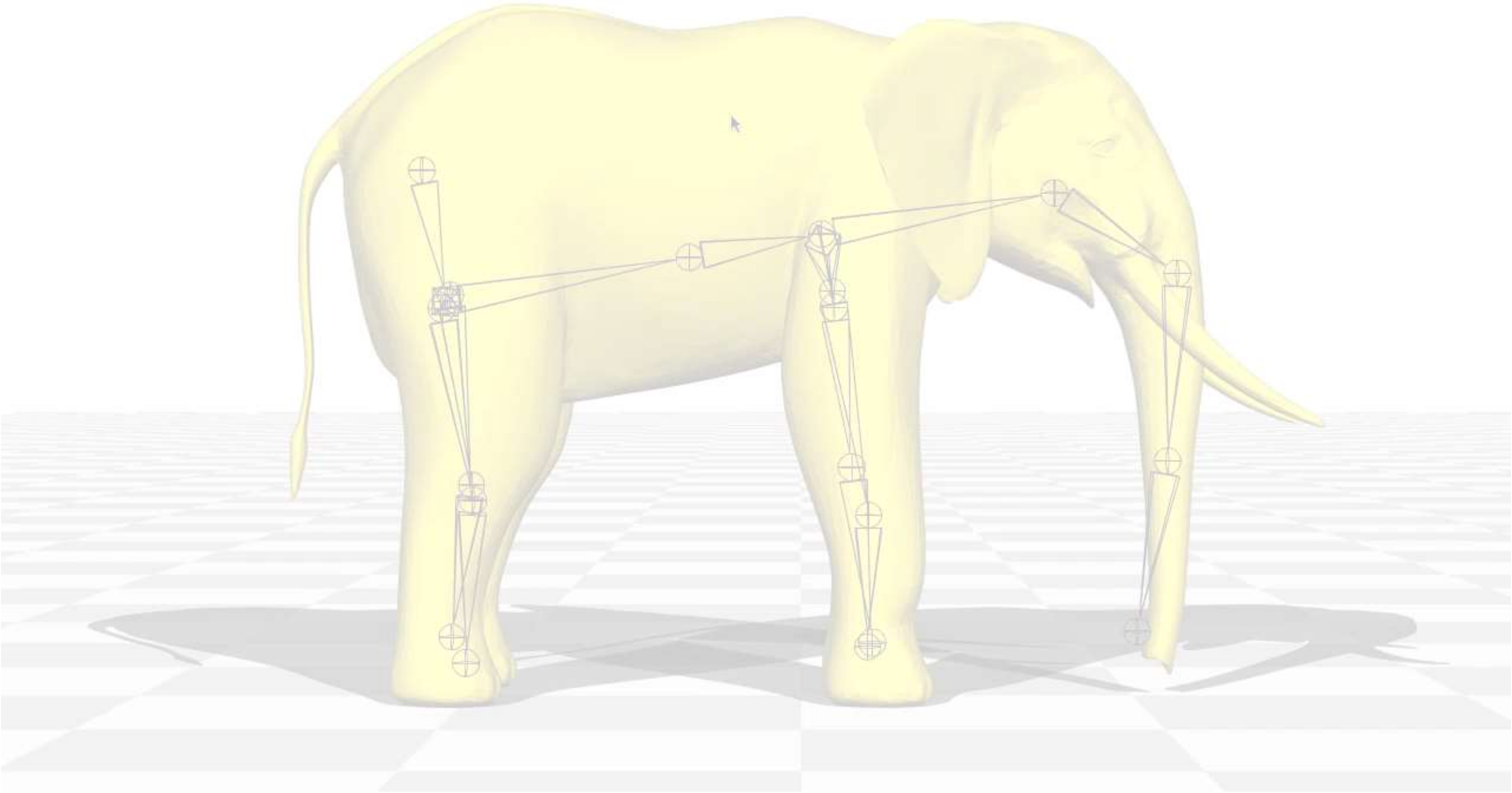
- Skinning, animation



http://youtu.be/P9fqm8vgdB8



http://youtu.be/Pjg33pH9RKo



Why C++?

- It is the industry standard for Computer Graphics
- It allows to write highly efficient code in a convenient way
- If you never used it before you will have to study it on your own
- The quality of the code will not be evaluated in the assignments. However, if you learn how to write good C++ code it will greatly simplify the homework

C++

- It is flexible, and many of the features are optional:
  - it can be used as an extension of C, with no objects
  - it can be used as a fully object oriented language
  - it has many advanced features such as “templates” that are useful to write efficient code that *is also readable*

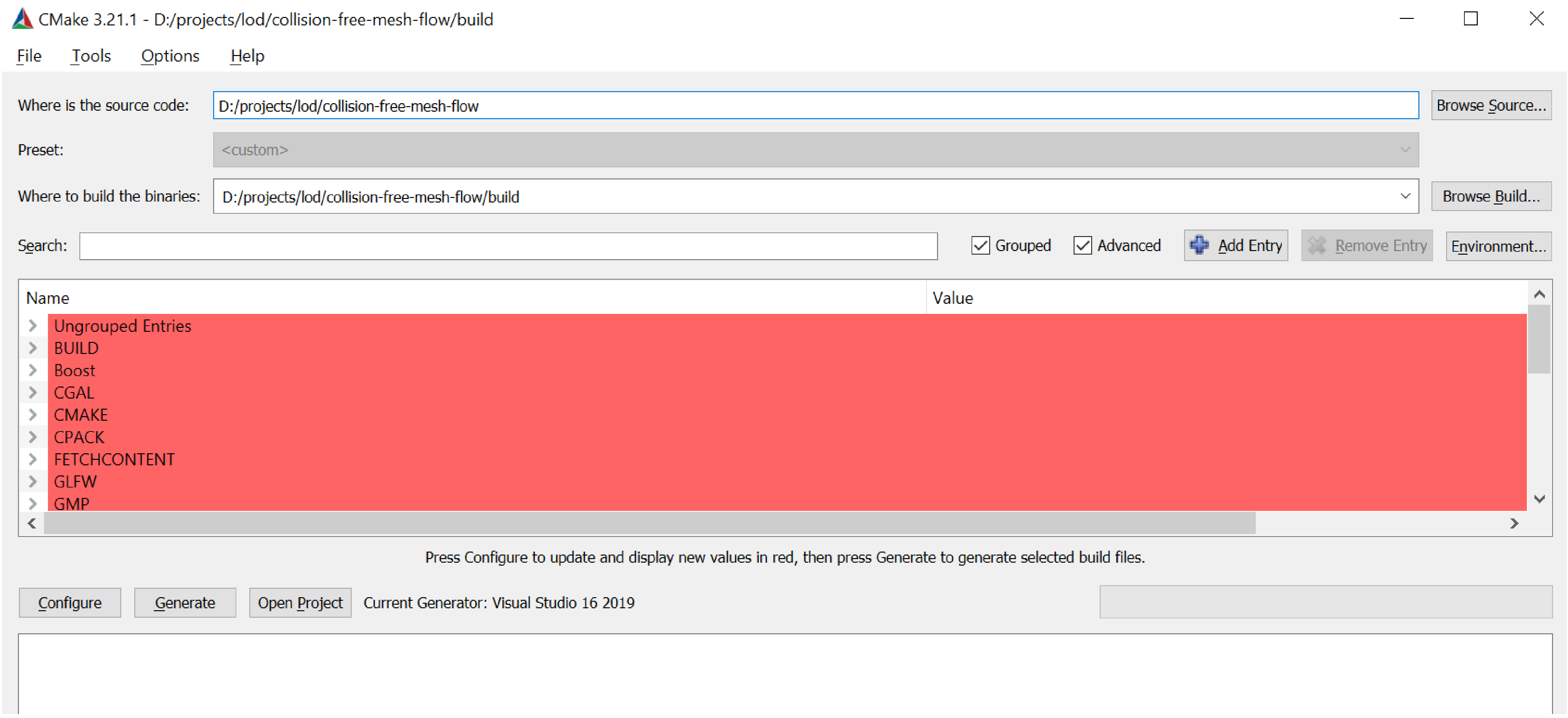
Comparison with Java

- **Java**: Everything must be placed in a class. **C++**: We can define functions and variables outside a class, using the same syntax used in ANSI C. The ”main“ function must be defined outside a class.
- **Java**: All user-defined types are classes. **C++**: We can define C types (enum,struct,array).
- **Java**: Single Inheritance. **C++**: Multiple Inheritance

Comparison with Java

- **Java**: No explicit pointers. **C++**: Explicit pointers and ”safe pointers“ (called Reference) are available.
- **Java**: Automatic memory management. **C++**: Manual memory management (like C) or semi-automatic management (shared pointers, http://en.cppreference.com/w/cpp/memory/shared_ptr).
- **Java**: All objects are allocated on the heap. **C++**: An object can be allocated on the heap or on the stack.

CMAKE



- build automation, testing, packaging and installation of software by using a compiler-independent method.
- cross-platform free and open-source software
- CMake is not a build system itself
- it generates another system's build files. It supports directory hierarchies and applications that depend on multiple libraries.
- It is used in conjunction with native build environments such as Make, Qt Creator, Ninja, Android Studio, Apple's Xcode, and Microsoft Visual Studio.
- It has minimal dependencies, requiring only a C++ compiler on its own build system.

https://cmake.org/cmake/help/latest/guide/tutorial/index.html

CMAKE

- If your project is in a folder Assignment_1, you need to:
  - mkdir build; cd build
  - cmake ../
- This will create the project. To compile it:
  - make (macosx/linux)
  - Open the project with visual studio on windows
- As an alternative, you can use Clion/Xcode/VS code that does all of this for you!

Basic CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.12)
project(Assignment1)

### Add src to the include directories
include_directories("${CMAKE_CURRENT_SOURCE_DIR}/src")

### Include Eigen for linear algebra
include_directories("${CMAKE_CURRENT_SOURCE_DIR}/../ext/eigen")

### Compile all the cpp files in src
file(GLOB SOURCES
"${CMAKE_CURRENT_SOURCE_DIR}/src/*.cpp"
)

add_executable(${PROJECT_NAME}_bin ${SOURCES})
```

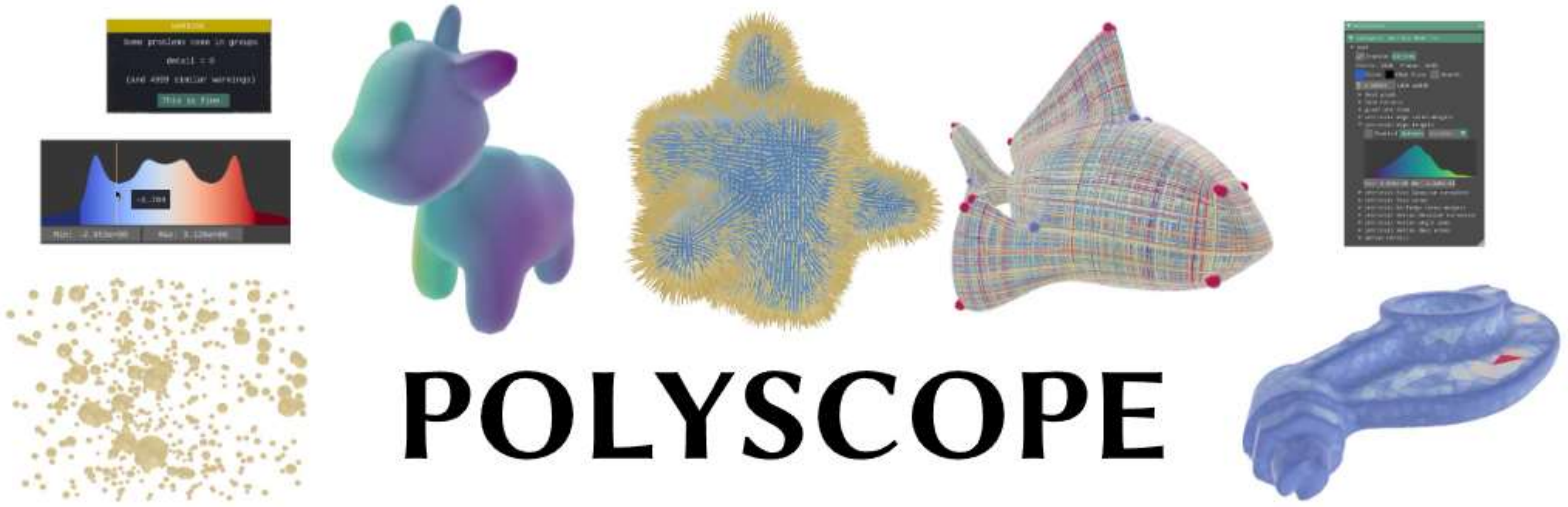
https://cmake.org/cmake-tutorial/

References

- **Thinking in C++ second edition** - Bruce Eckel
  http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html
- **Wikipedia** http://en.wikipedia.org/wiki/C%2B%2B
- **Cpp Reference** http://www.cppreference.com
- **Cmake** https://cmake.org

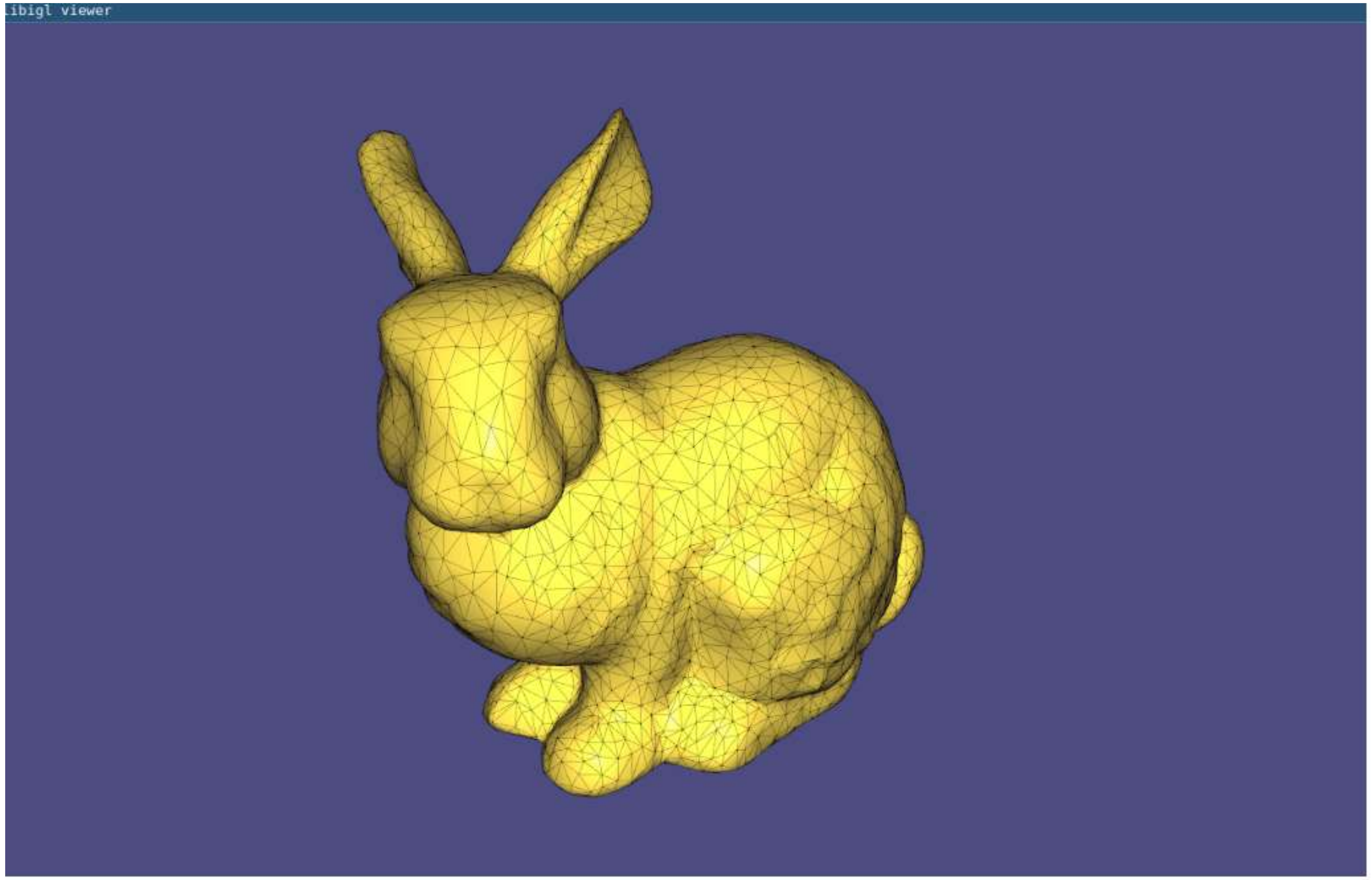
Code bases for your homework

https://libigl.github.io/tutorial/



http://polyscope.run/

Homework for next class



(Example 102) loads and draws a mesh.

Feedback?