Shape Modeling and Geometry Processing
Assignment 1 - libigl “Hello world”

Aviv Segall

INTERACTIVE GEOMETRY LAB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Exercise Sessions

- Friday from 11:15-12:00
- Exercise presentation + Q&A + live demo
- Course website: http://igl.ethz.ch/teaching/shape-modeling/sm2025/
- Assistants: Annika Oehri, Alexandre Binninger, Aviv Segall

Rules of the Game

- Total of 5 mandatory + 1 optional assignments
  - Each to be completed in 2-3 weeks' time
- Deliverables: code + live demo (sometimes a mini-report)
  - Graded on all mandatory assignments
  - Instructions given per assignment
- Submit until 10:00 on the day of the due date
- Late submissions not accepted, **strict policy**.

Rules of the Game - live demo

- We tell you what to show, in-person
- During the exercise session slot
- 3-4 minutes per person, schedule announce beforehand
- We might ask you to change the mesh or parameters

Rules of the Game

- One mini-exam
  - multiple-choice questions
  - 60 minutes
  - Wed 28.05.2025, 10:00, during the lecture time

Rules of the Game

- Grading = Exercises (80%) + mini-exam (20%), 8 credits
  - Assignments are weighed differently

	ex1	ex2	ex3	ex4	ex5	ex6	mini-exam
Weight/Points	10	16	Optional	16	16	22	20

- Zero points for copied code or report
- Be fair, be honest and have fun
  - Do not make your code public, **even after the semester ends!**

Rules of the Game

DO NOT COPY

- We will run a plagiarism check on all the assignments
  - Including ones from previous years
  - Plagiarism could lead to expulsion from the course

Asking questions

- Through Github issues (on the template repo)
  - Do not use e-mails to directly ask questions to TAs
  - Please check if your question has already been asked before posting
  - Feel free to answer questions by other students!
  - **Don’t wait until the last minute with problems**

Asking questions

- Q&A sessions during TA slots.
  - Presence not mandatory, content not necessary for assignments.
  - Use it if you need to show your code…
  - … but remember: you are expected to understand your algorithms and debug your code yourself.

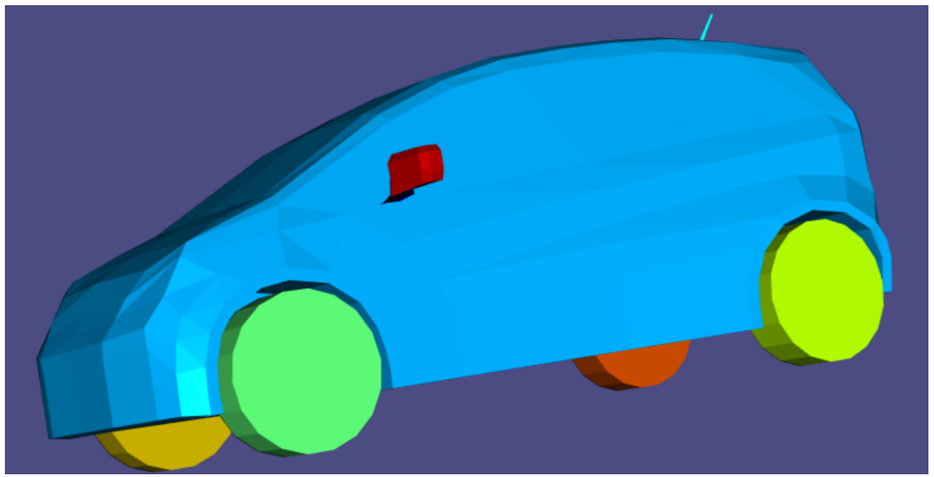
Asking questions

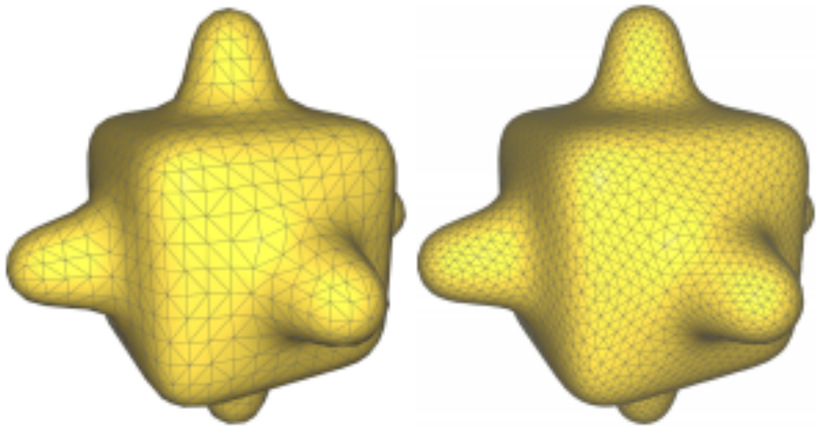
- Any emergencies or personal concerns:
  - e-mail at igl.lectures@inf.ethz.ch

Assignment Overview

ex	Title	Main TA	Start	End
1	**Mesh "Hello World"**	**Aviv Segall**	21.02.2025	07.03.2025
2	**Surface reconstruction**	**Alexandre Binninger**	07.03.2025	28.03.2025
3	**Discrete differential geometry**	**Alexandre Binninger**	21.03.2025	N/A
4	**Mesh parameterization**	**Annika Oehri**	28.03.2025	02.05.2025
5	**Shape deformation**	**Annika Oehri**	02.05.2025	23.05.2025
6	**Skinning & deformation**	**Aviv Segall**	16.05.2025	13.06.2025

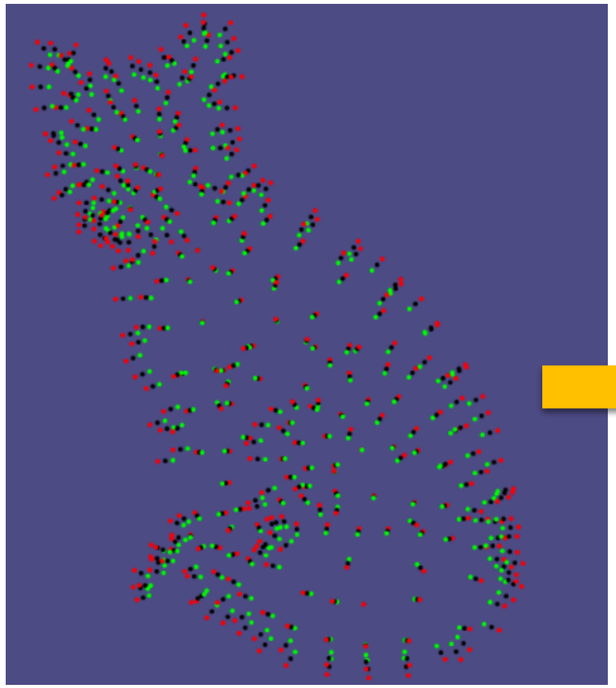
Ex1 Mesh “Hello World!”



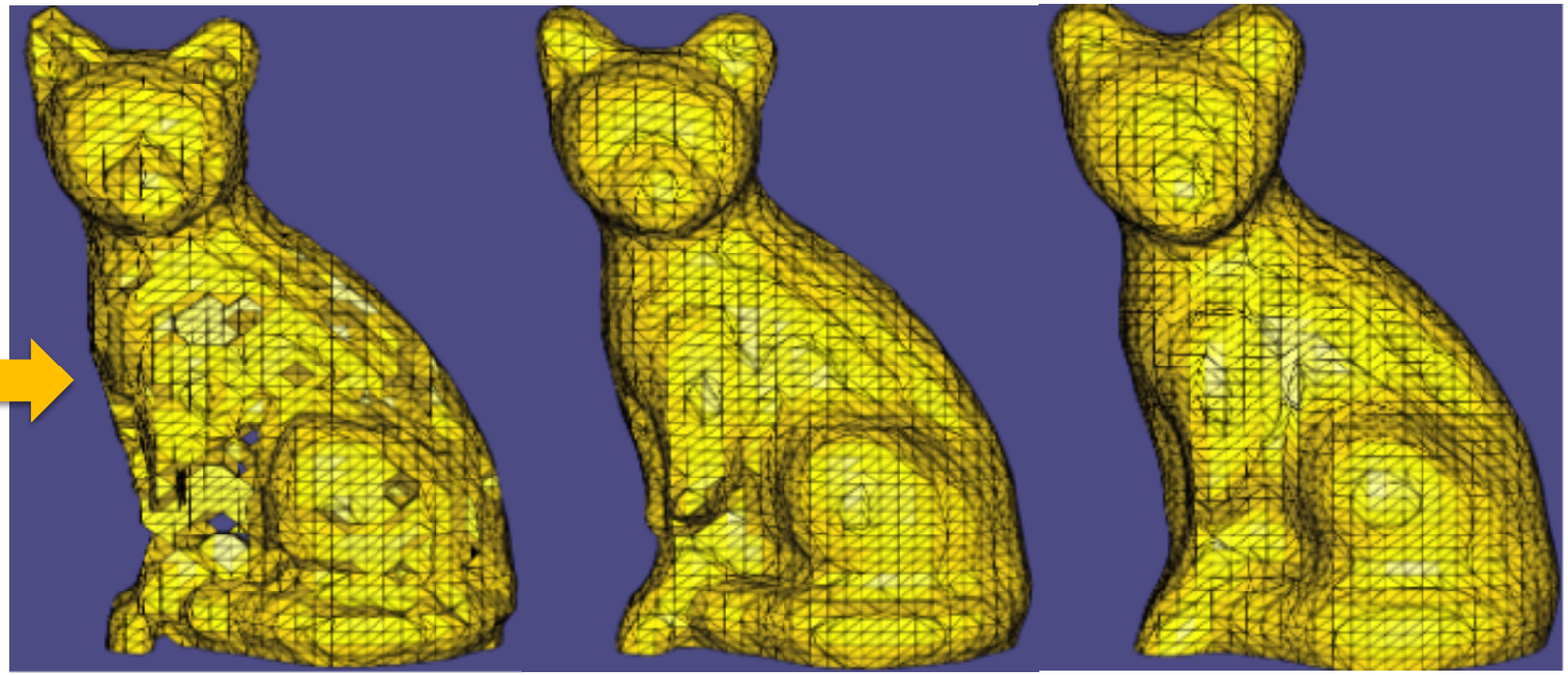


Set up the environment and get familiar with libigl

Ex2 Surface Reconstruction

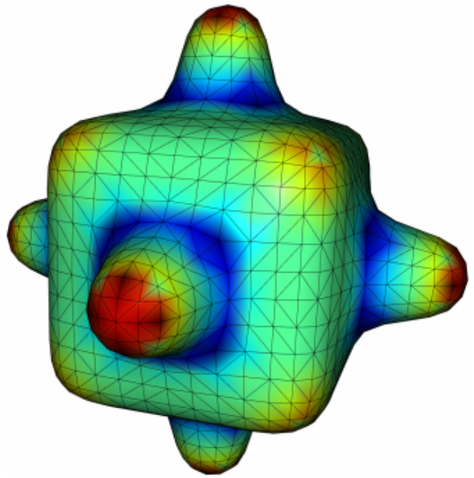


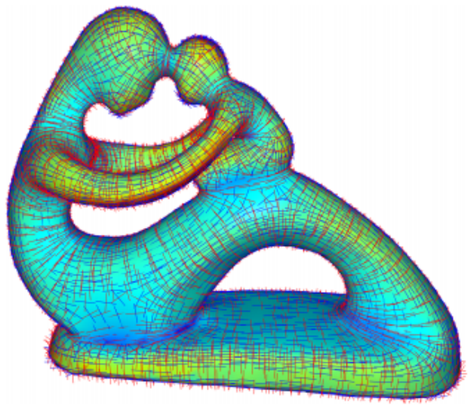
Input 3D points



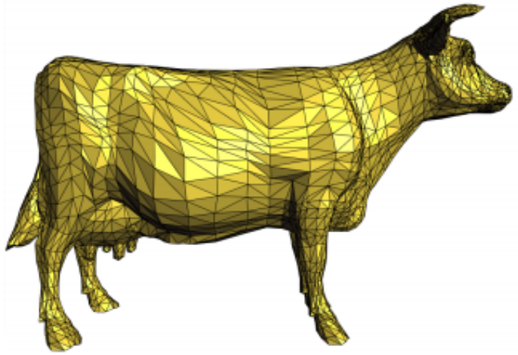
3D meshes reconstructed from input
using different parameters

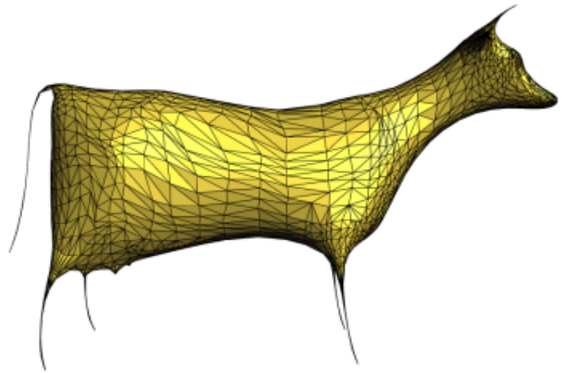
Ex3 discrete differential geometry





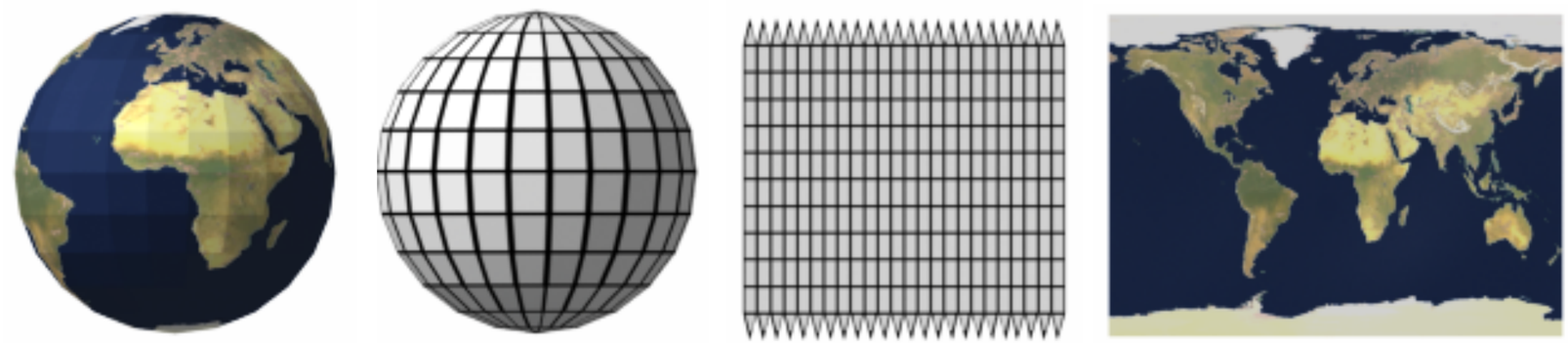
Curvature

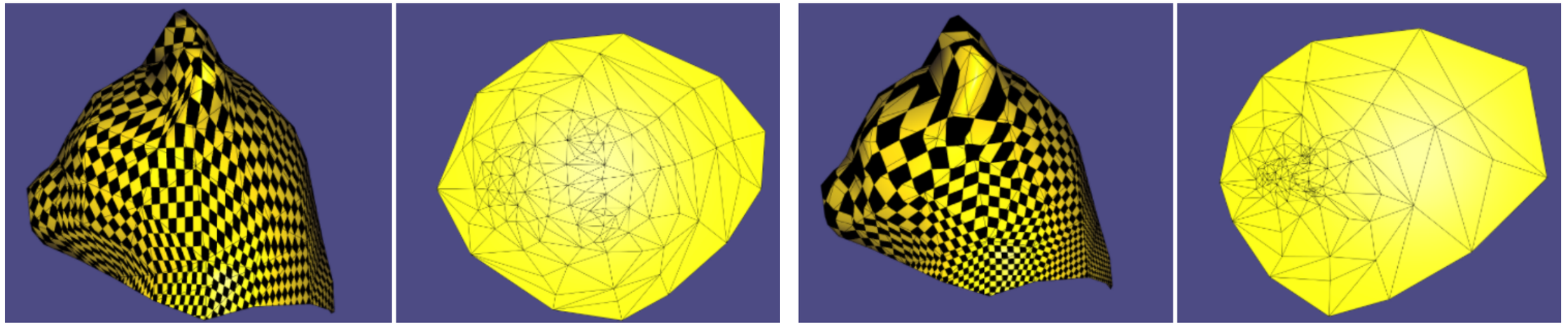




Smoothing

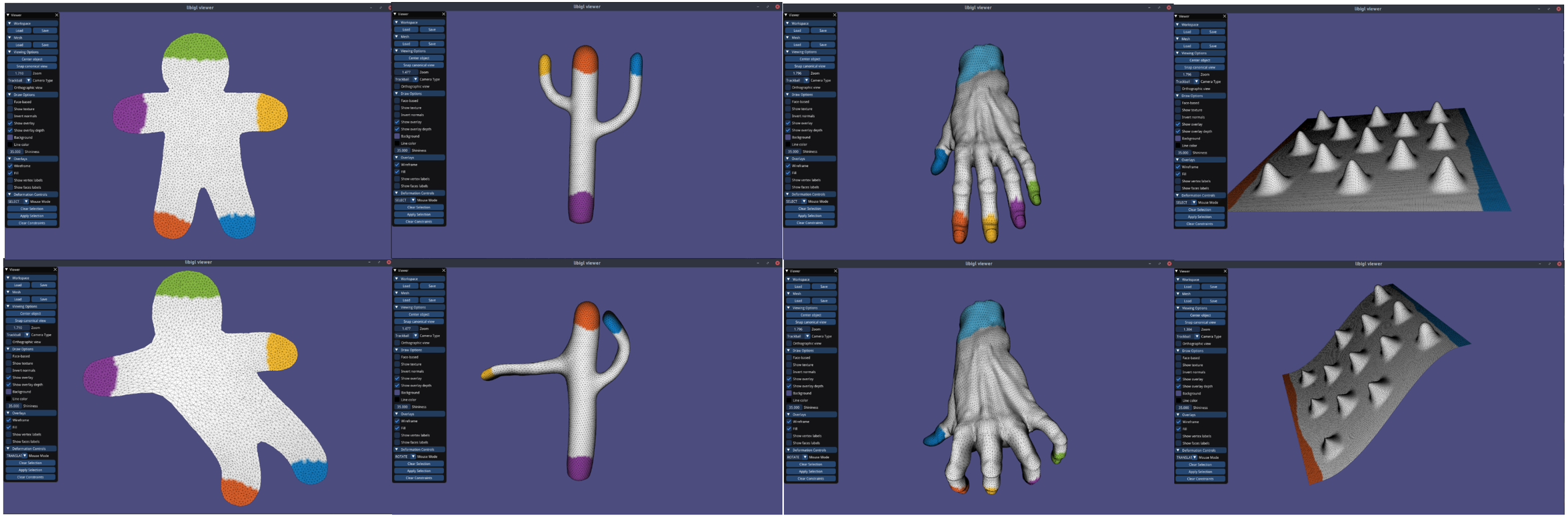
Ex4 Mesh parameterization





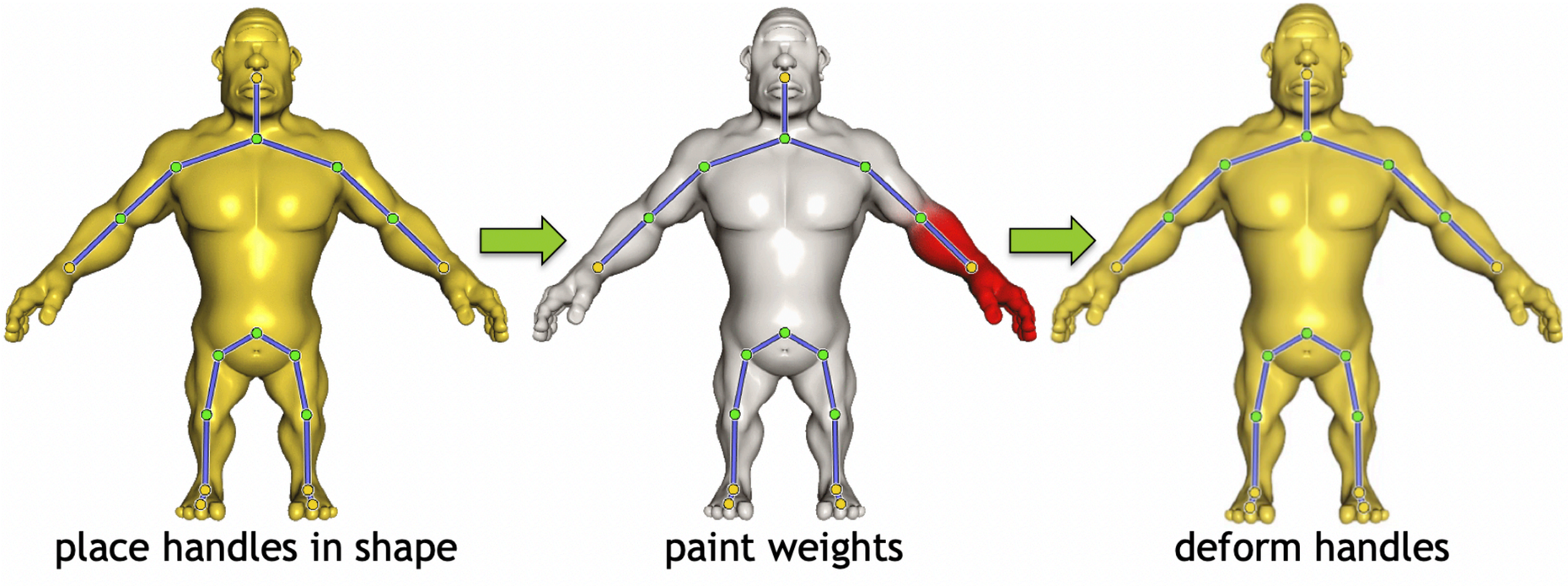
Mapping 2D textures to 3D surfaces

Ex5 Shape deformation



Implement a UI to perform shape deformation

Ex6 Skeletal animation

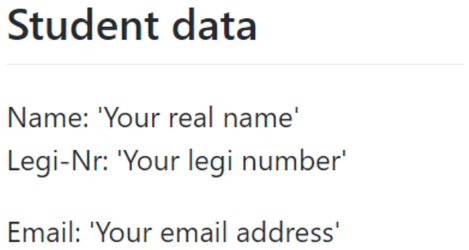


Assignments on Git

- We use Git classroom, see instructions: https://github.com/eth-igl/GP2005-Assignments
- Reports should be placed in Readme.md of each assignment and **nicely typeset.**
- You can use STL & Eigen but no non-standard dependencies.

Assignments on Git

- Add your name and email address



- Commit and push your code frequently
- Regularly check the main repository for updates and new instruction:
  - https://github.com/eth-igl/GP2025-Assignments
  - Update the new assignments manually if ‘git pull base main’ doesn’t work for you.

Programming Tips

- /!\ Compile in **Debug** mode to get more info on errors, compile in **Release** mode for better speed
- This can make a huge difference for interactive problems that should run in real-time (e.g., assignment 5)

Programming Tips - demo

- We will check your code in detail if we have doubts about your results.
- Remove unnecessary print statements from your final submission. They slow things down.
  - We will subtract points for unnecessary slow demos.
- Use command line/GUI parameters (e.g. to change a mesh).
  Don’t hardcode & recompile to change it!

Supported OS

- We offer support on Windows and MacOS if you have problems compiling, especially at the beginning.
- For other OS, it should work fine as well, but it is your responsibility.
  - Exception: WSL likely **won't** work

Exercise 1 - libigl “Hello World!”

libigl - A simple C++ geometry processing library

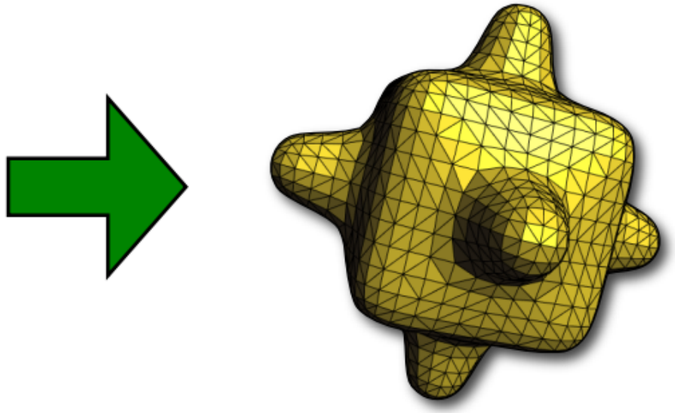
build passing build passing

Exercise 1 - libigl “Hello World!”

- Read and display a mesh

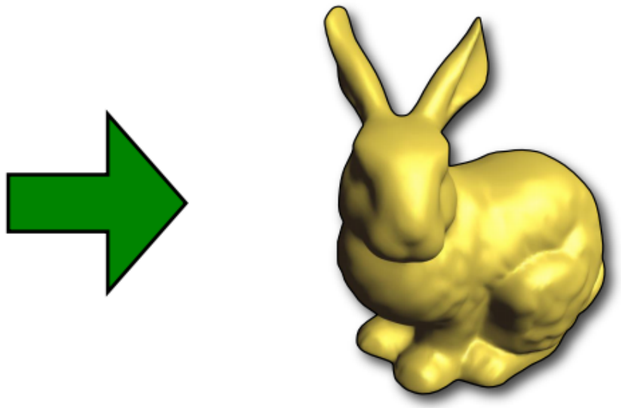
```
OFF
1250 2496 0
-2.09105 -2.09105 2.09105
-0.833333 -2.23958 2.23958
0.833333 -2.23958 2.23958
2.09105 -2.09105 2.09105
...
3 940 83 320
3 386 0 941
...
```

bumpy_cube.off



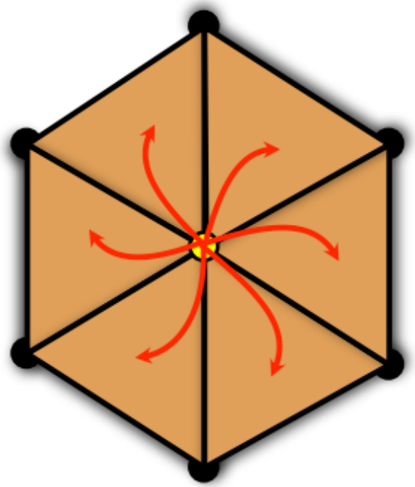
```
# Wavefront OBJ file
v 30.50959969 12.17459898 -15.84426970
v 30.49857998 11.87718728 -15.40759913
v 30.53679943 12.68500615 -14.82485356
v 30.67168999 11.71161003 -15.78844530
...
f  633/16706 11590/29979 4339/16704
f  11590/3161 633/16716 19901/16699
...
```

bunny.obj

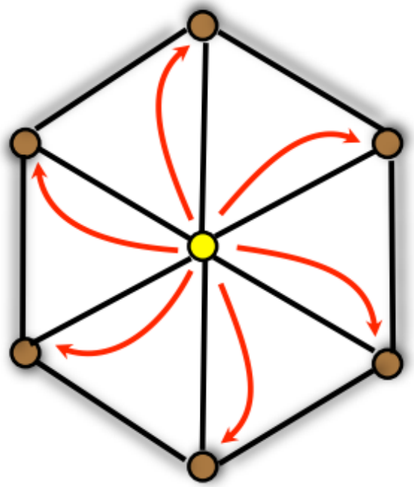


Exercise 1 - libigl “Hello World!”

- Perform simple neighborhood calculations



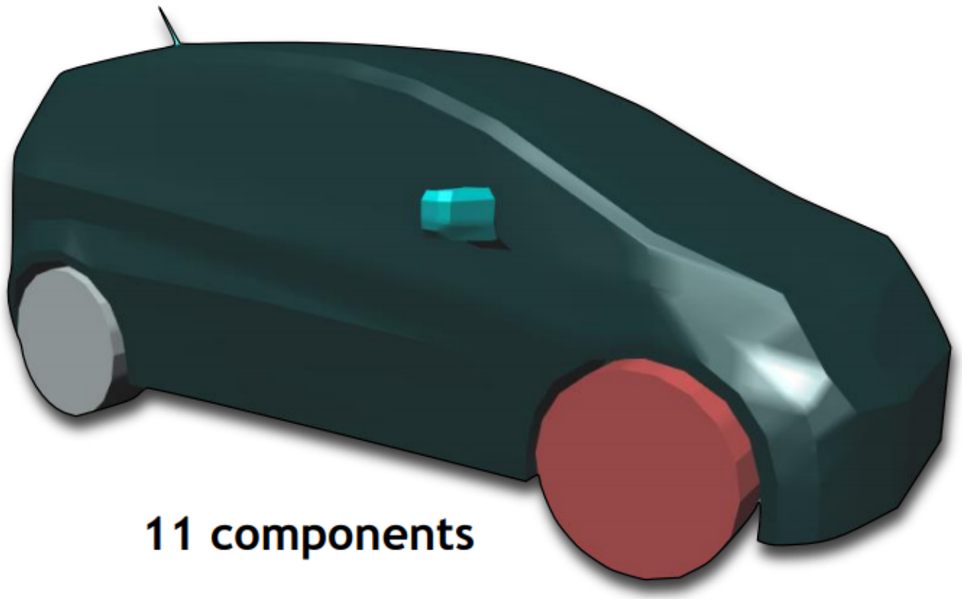
vertex-to-face



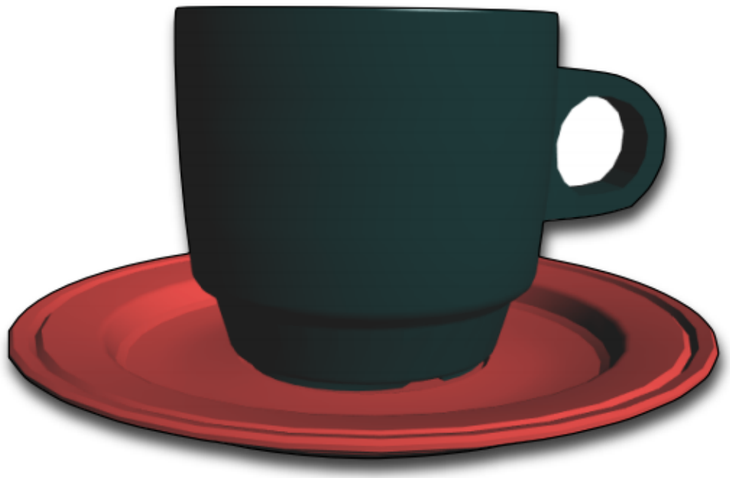
vertex-to-vertex

Exercise 1 - libigl “Hello World!”

- Connected components



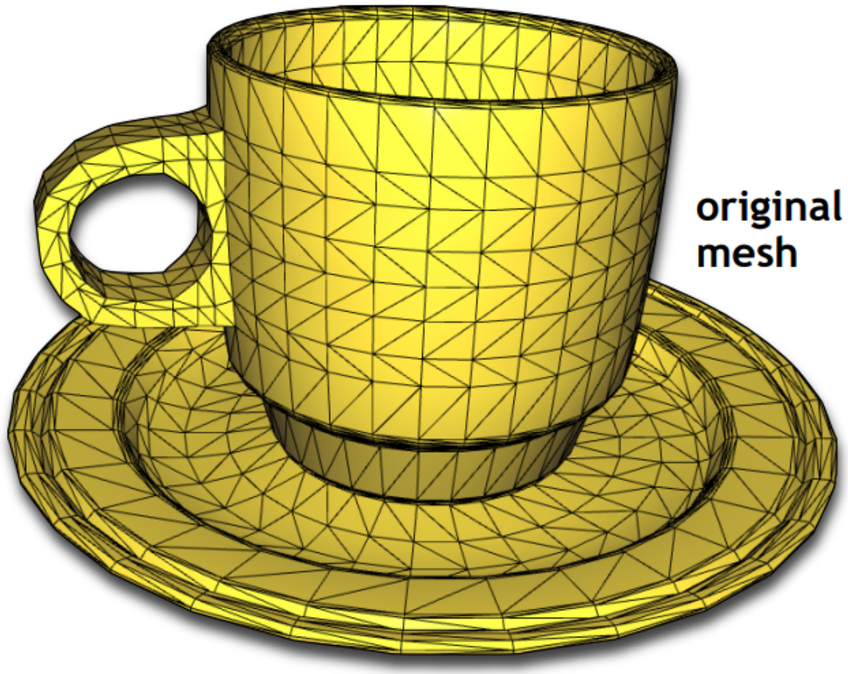
11 components

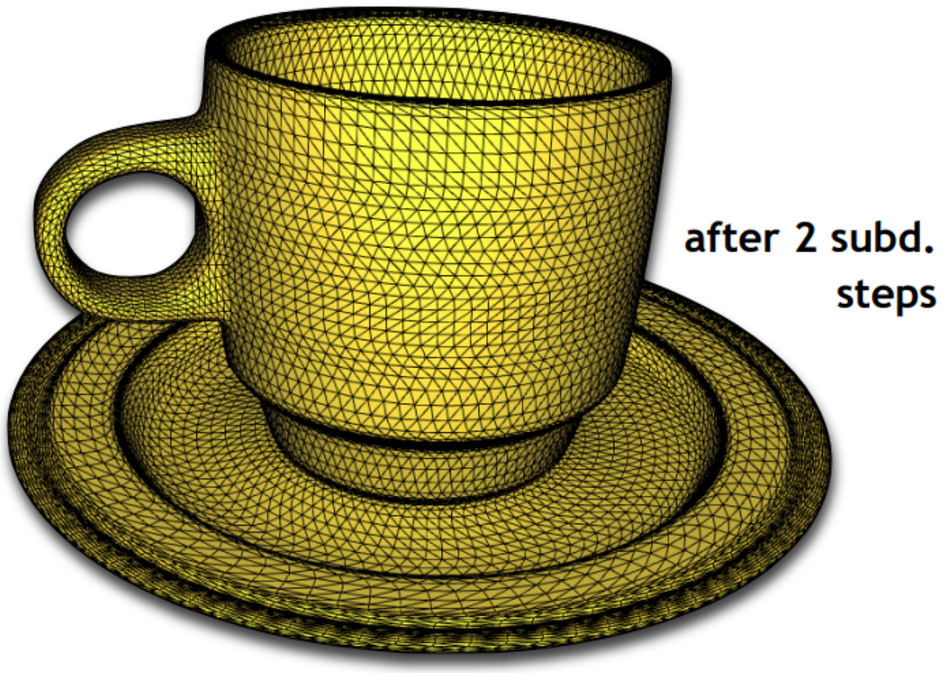


2 components

Exercise 1 - libigl “Hello World!”

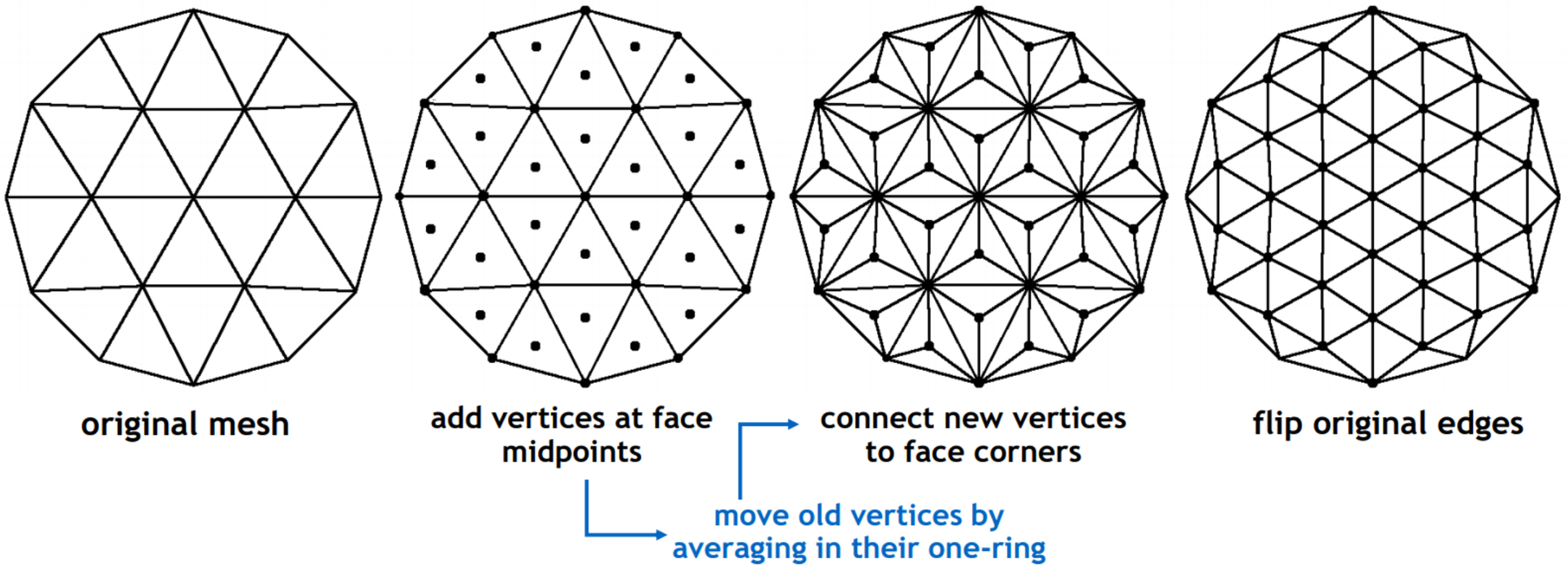
- $\sqrt{3}$ subdivision





Exercise 1 - libigl “Hello World!”

- $\sqrt{3}$ subdivision



Eigen

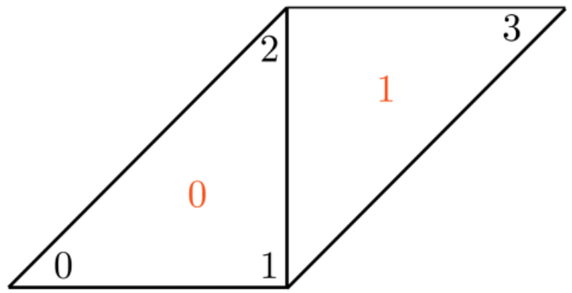
- A C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms
  - http://eigen.tuxfamily.org/
- Header-only library
  - No compilation required!
- Tutorials:
  - http://eigen.tuxfamily.org/dox/group__TutorialMatrixClass.html
  - http://eigen.tuxfamily.org/dox/group__QuickRefPage.html
  - **IMPORTANT:** https://eigen.tuxfamily.org/dox/TopicPitfalls.html

Mesh Representation with Eigen

- An Eigen matrix

```
Eigen::Matrix< type, #rows, #cols>
```

$$V = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 3 & 2 \end{pmatrix}$$




```
Eigen::Matrix<double, Eigen::Dynamic, 3> V;
Eigen::Matrix<int,    Eigen::Dynamic, 3> F;
```

- Have a look at: http://eigen.tuxfamily.org/dox/GettingStarted.html

Eigen common pitfalls

Aliasing issues:

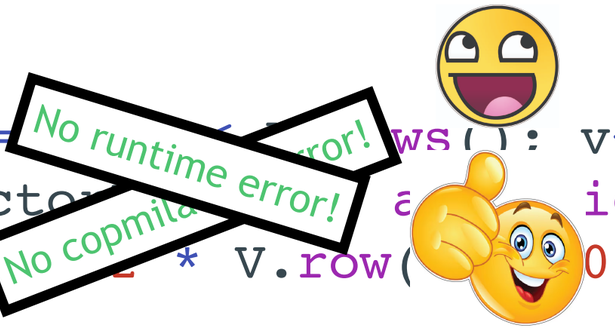
```
Eigen::Matrix3d a;
a << ...
a = a.transpose();        // WRONG!
a = a + a.transpose();    // WRONG!
a = a * a; // That's actually fine
```

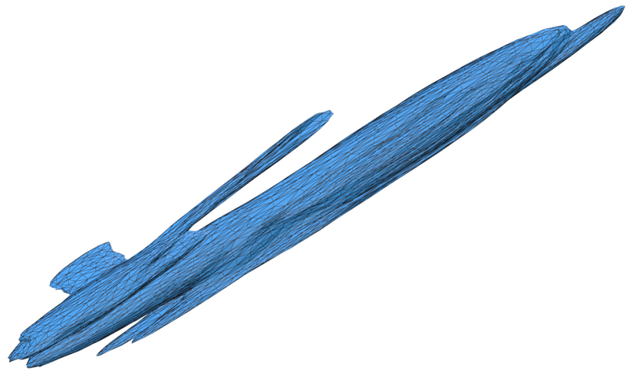
Solution:

```
Eigen::Matrix3d a;
a << ...
a.transposeInPlace();
a = (a + a.transpose()).eval();
```

Is this code correct?

```
for (int v = 0; v < V.rows(); v++) {
  Eigen::VectorXi a = adj.neighbors(v);
  V.row(v) = 0.2 * V.row(v) + 0.8 * avg;
}
```

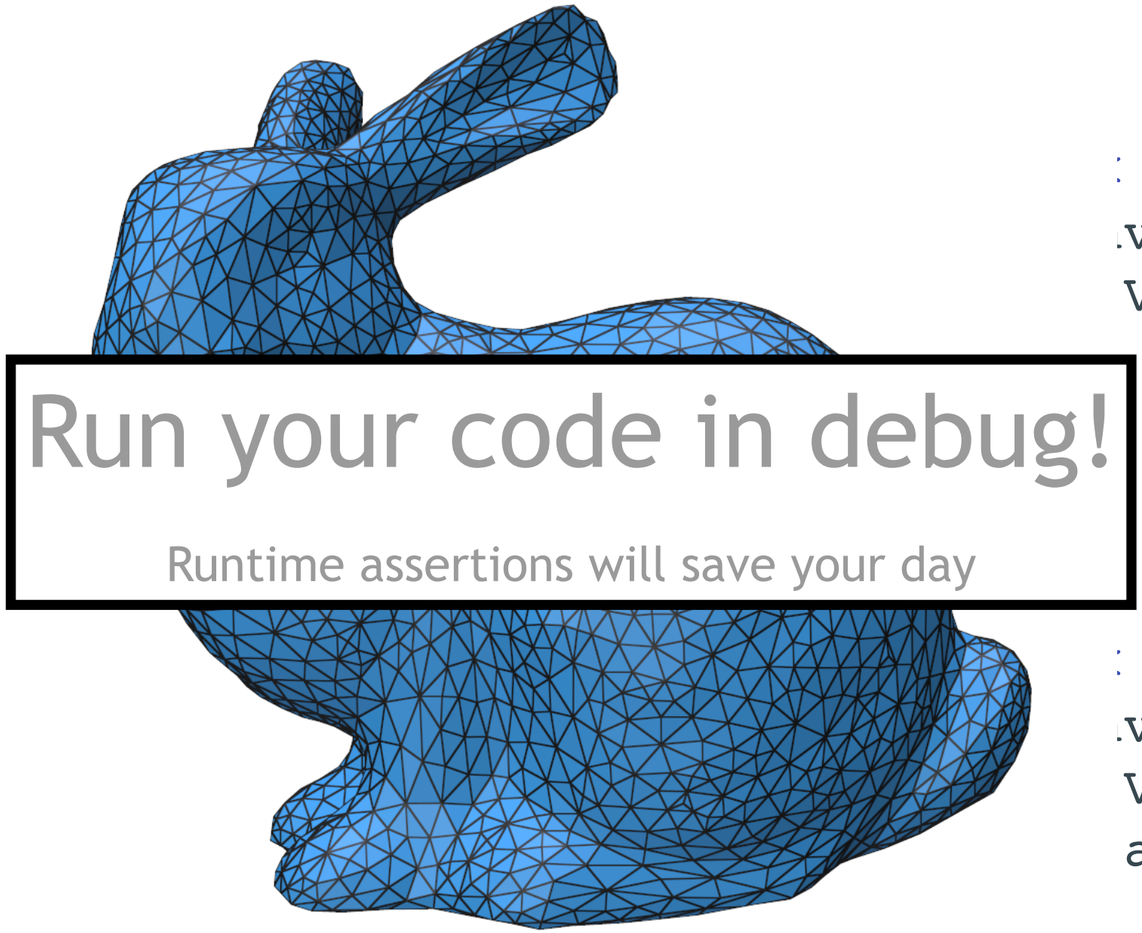




Eigen common pitfalls

Aliasing issues:

```
Eigen::Matrix3d a;
a << ...
a = a.transpose();
a = a + a.transpose(
a = a * a; // That's
```



Run your code in debug!

Runtime assertions will save your day

```
: V.rows(); v++) {
.vg = avg_neighbors(v);
V.row(v) + 0.8 * avg;
```

Solution:

```
Eigen::Matrix3d a;
a << ...
a.transposeInPlace()
a = (a + a.transpose
```

```
: V.rows(); v++) {
.vg = avg_neighbors(v);
V.row(v) +
avg.transpose();
```

Libigl

- Open source C++ library for geometry processing
  - Minimal dependencies
  - Header-only
  - No complex data types
- Dependencies: Eigen, GLFW, imgui, ...
  - All included as submodules in libigl
- Tutorials that explain the main functionality
  - Compilation instructions: see assignment sheet
- **Header files contain documentation for functions**

Libigl – The Viewer

- Display mesh
- Very basic UI
- More options in code

- Demo tutorial 106

Adding functionality to the viewer

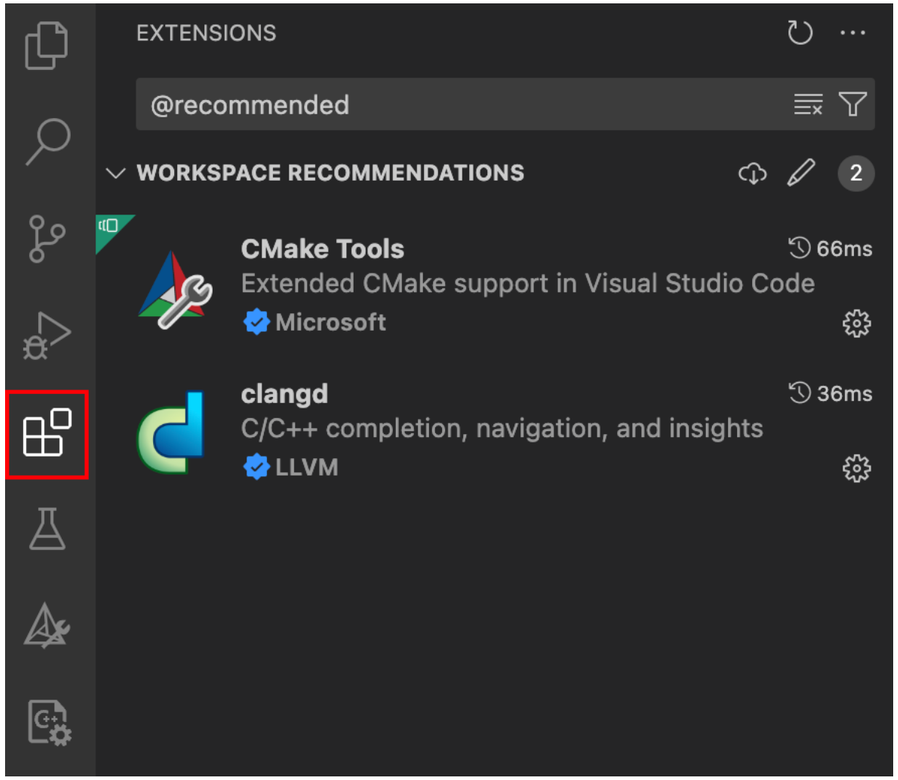
- Custom callbacks for keyboard/mouse interactions supported
  - See tutorial 103_Events
- Also supported: Face/Vertex Colors, Overlays (points/lines)
  - See tutorials 104_Colors, 105_Overlays
- See https://libigl.github.io/tutorial/

Tasks for exercise 1

- Compiles a single main.cpp that runs the viewer
  - Assignment tasks are to be implemented as key interactions
  - ‘1’ - ‘2’ : neighborhood relations
  - ‘3’ - ‘5’: shading
  - ‘6’: connected components
  - ‘7’: subdivision

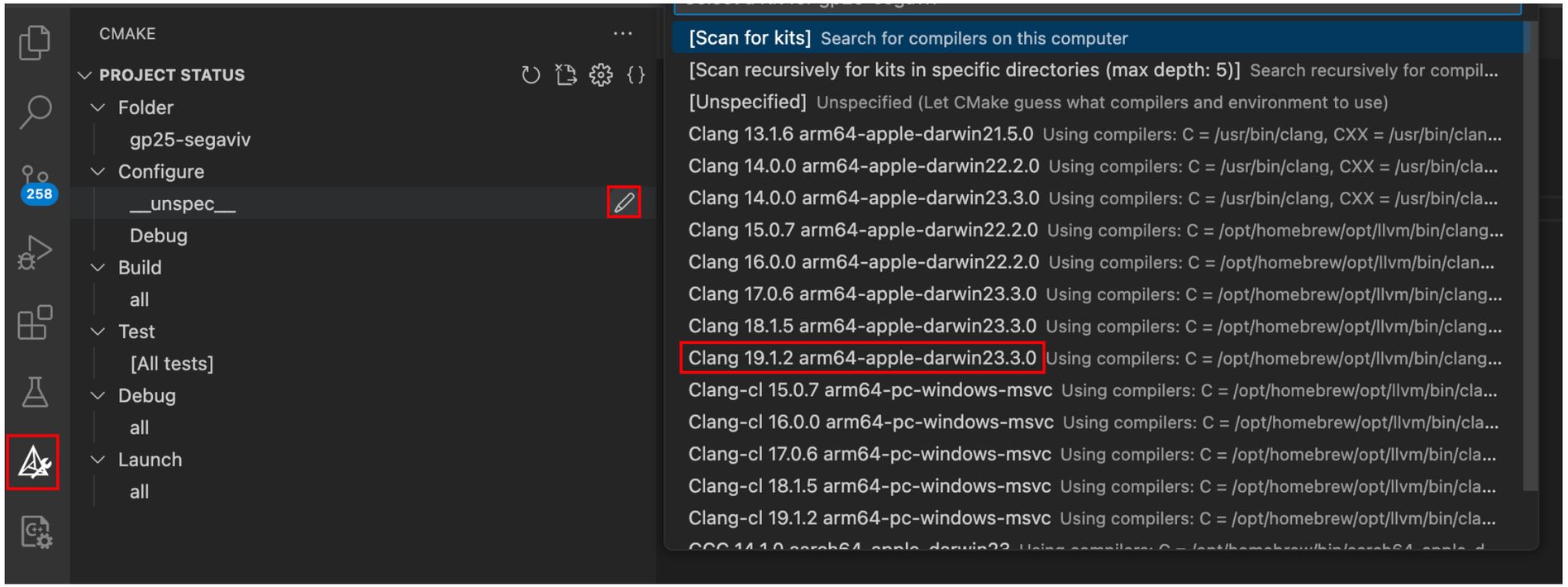
IDE tips

- Use Visual Studio Code
- Useful extensions:
  - cmake tools: UI for cmake
  - clangd: better syntax analysis



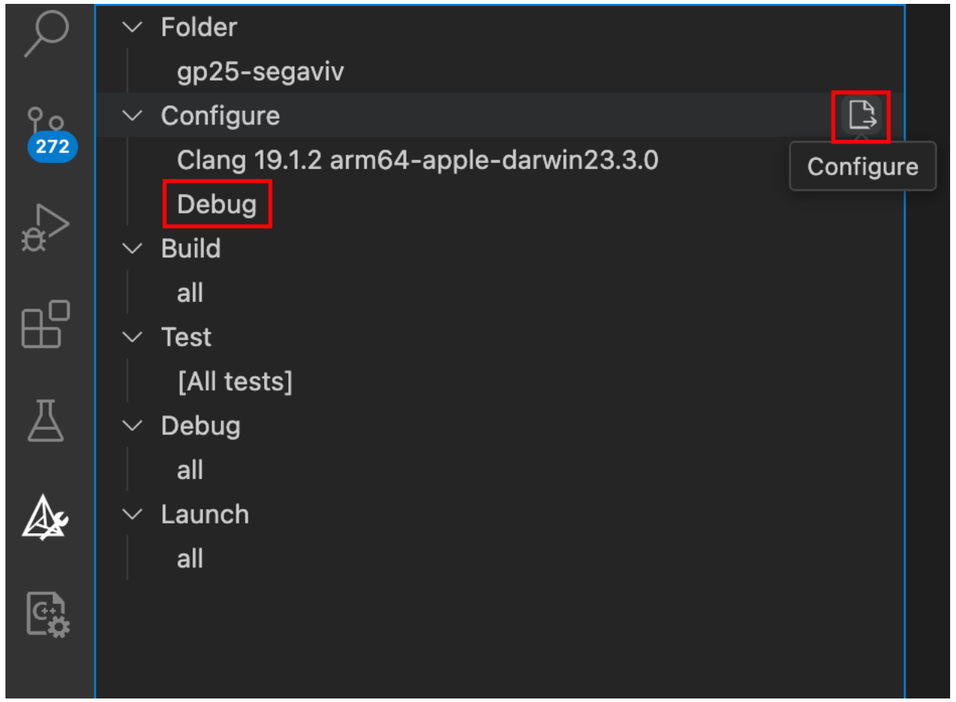
IDE tips

- CMake Tools
  - Choose kit (use clang on Mac, **not gcc**)



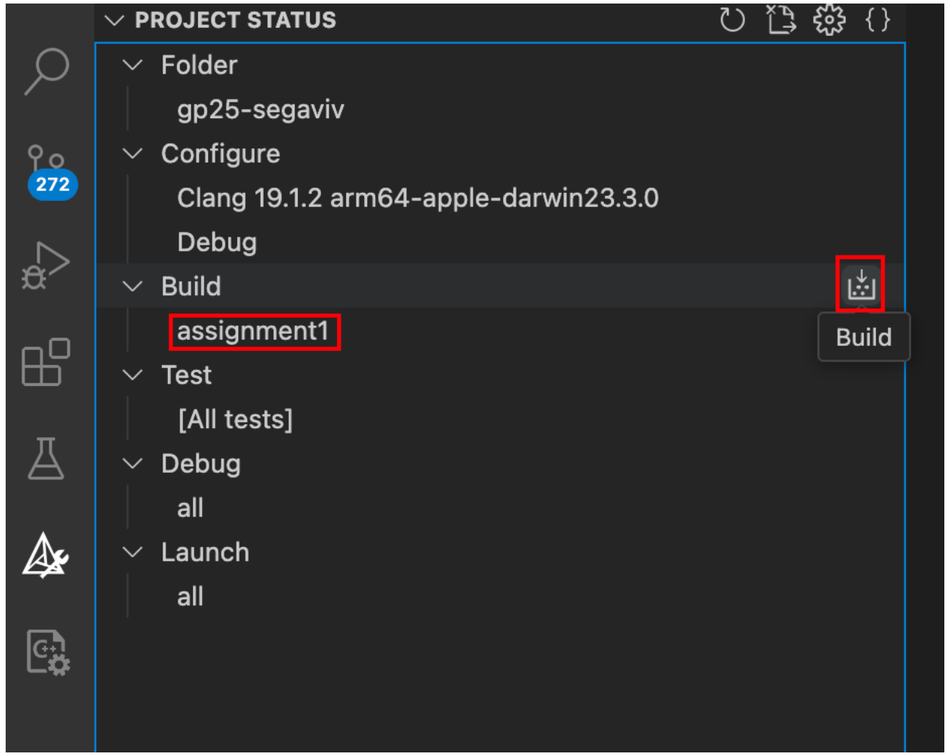
IDE tips

- CMake Tools
  - Choose kit (use clang on Mac, **not gcc**)
  - Configure



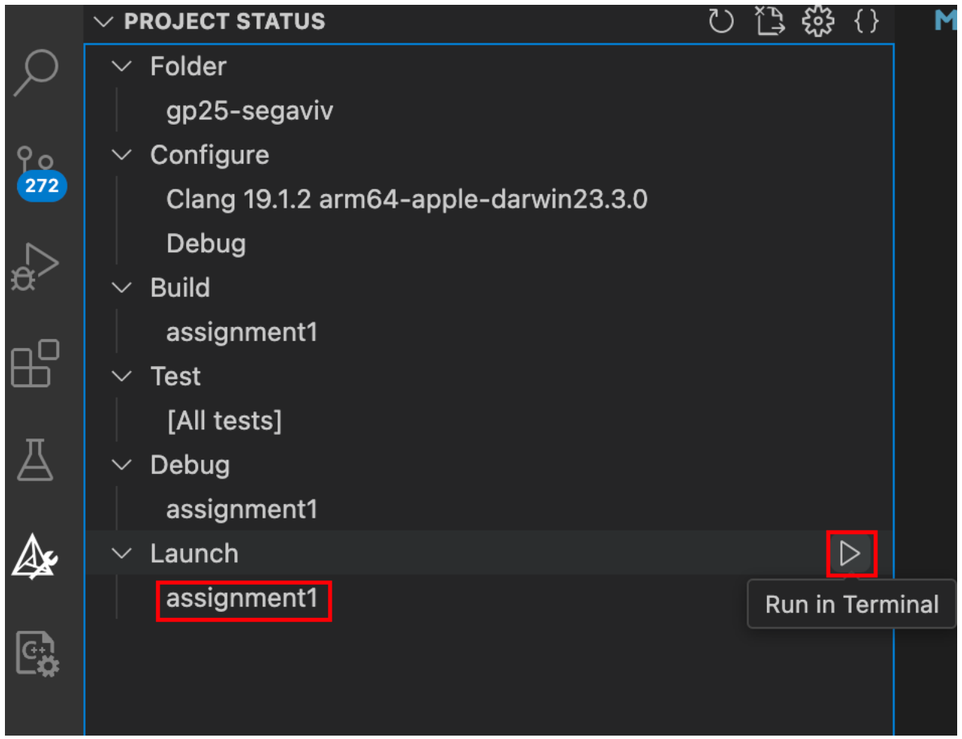
IDE tips

- CMake Tools
  - Choose kit (use clang on Mac, **not gcc**)
  - Configure
  - Build



IDE tips

- CMake Tools
  - Choose kit (use clang on Mac, **not gcc**)
  - Configure
  - Build
  - Run/Debug



IDE tips

- CMake Tools - suggested setup

Separate folder for Release/Debug (avoid recompiling everything when switching configuration)

.vscode > {} settings.json > ...

```
1  {
2    "cmake.buildDirectory": "${workspaceFolder}/build/${buildType}",
3    "cmake.copyCompileCommands": "${workspaceFolder}/build/compile_commands.json",
4    "cmake.debugConfig": {
5      "args": ["${workspaceFolder}/assignment1/data/bunny.off"]
6    }
7  }
```

Thank you!