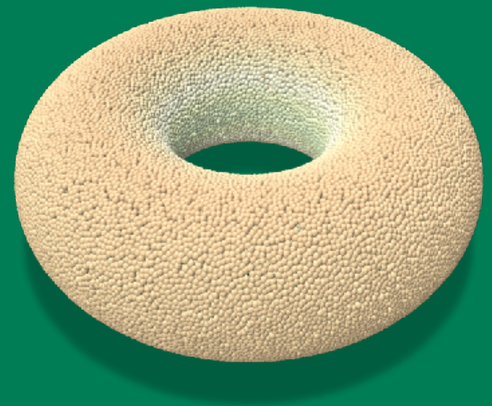
Exploring the Laplacian in Computer Graphics

Week 3



Crane He Chen

The Johns Hopkins University

2023 Fall

Today's Schedule

- Lecture about CMake
- Show some demos about how to use CMake
- Pick up from where we dropped last week, the python pair-coding

Format of Future Lectures

- C++ pair-coding is moved to next week
- Starting next week, each 75 lecture = 60-minute theory + 15 minute pair coding
- By the time of art-contest, everyone will succeed in C++
- Folks who figured out fast, can use the pair-coding opportunity to run more demos that do funky things (e.g. swept volumes / cubic stylization)

Coding Fundamentals for This Course

How do you run a C++ code

High level concept:

Run in terminal/prompt

these commands stay

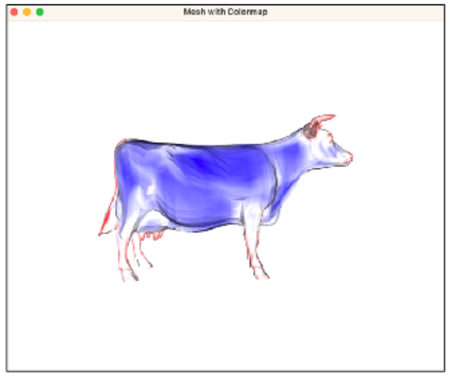
```
mkdir build
cd build
cmake ..
make
```

Your golden hammer!

this command change

```
./Draw
```

usually, you want to
visualize result + save result to a file



Your GUI
(graphical user interface) !

What is CMake

- A build manager for C++
- Helps you find C++ packages installed in the system
- Platform independent, works for MacOS, Windows, Ubuntu

Why do we love CMake?

Compilation of a C++ project is frustrating before CMake came into existence!

What? This codebase only compiles on Linux?

How do I write this thing called “Makefile” to compile my code? Why is there always a library that fails to be linked?

What is CMake

Before CMake came into existence, you need to write “Makefile” (~1k lines)

Makefile

Makefile

Users > crane > Downloads > curvature-qslim-mesh-decimation > build > Makefile

```
68  # Special rule for the target edit_cache
69  edit_cache:
70      @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running CMake cache editor..."
71      /usr/local/Cellar/cmake/3.26.3/bin/ccmake -S$(CMAKE_SOURCE_DIR) -B$(CMAKE_BINARY_DIR)
72  .PHONY : edit_cache
73
74  # Special rule for the target edit_cache
75  edit_cache/fast: edit_cache
76  .PHONY : edit_cache/fast
77
78  # Special rule for the target rebuild_cache
79  rebuild_cache:
80      @$(CMAKE_COMMAND) -E cmake_echo_color --switch=$(COLOR) --cyan "Running CMake to regenerate build sy
81      /usr/local/Cellar/cmake/3.26.3/bin/cmake --regenerate-during-build -S$(CMAKE_SOURCE_DIR) -B$(CMAKE_
82  .PHONY : rebuild_cache
83
84  # Special rule for the target rebuild_cache
85  rebuild_cache/fast: rebuild_cache
86  .PHONY : rebuild_cache/fast
87
88  # The main all target
89  all: cmake_check_build_system
```

What is CMake

Now you just need to write CMakeLists (~30 lines)!

CMake will help you generate Makefile automatically!

CMakeLists.txt

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)
project(TotalCurvature)

list(PREPEND CMAKE_MODULE_PATH ${CMAKE_CURRENT_SOURCE_DIR}/cmake)

# Eigen
if(WIN32)
    # On Windows, include Eigen from the local source directory
    include_directories(${CMAKE_CURRENT_SOURCE_DIR}/eigen)
else()
    # On other operating systems, use find_package
    find_package(Eigen3 REQUIRED)
    include_directories(${EIGEN3_INCLUDE_DIR})
endif()

# Libigl
include(libigl)

# Enable the targets
igl_include(core)
igl_include(copyleft cgal)

# polyscope
add_subdirectory("polyscope")

# openmp
find_package(OpenMP REQUIRED)

# Add your project files
file(GLOB SRC_FILES main.cpp)
add_executable(TotalCurvature ${SRC_FILES})
target_link_libraries(TotalCurvature PUBLIC igl_copyleft::cgal polyscope PRIVATE OpenMP::OpenMP_CXX)
```

What is CMake

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)
project(TotalCurvature)

list(PREPEND CMAKE_MODULE_PATH ${CMAKE_CURRENT_SOURCE_DIR}/cmake)

# Eigen
if(WIN32)
    # On Windows, include Eigen from the local source directory
    include_directories(${CMAKE_CURRENT_SOURCE_DIR}/eigen)
else()
    # On other operating systems, use find_package
    find_package(Eigen3 REQUIRED)
    include_directories(${EIGEN3_INCLUDE_DIR})
endif()

# Libigl
include(libigl)

# Enable the targets
igl_include(core)
igl_include(copyleft cgal)

# polyscope
add_subdirectory("polyscope")

# openmp
find_package(OpenMP REQUIRED)

# Add your project files
file(GLOB SRC_FILES main.cpp)
add_executable(TotalCurvature ${SRC_FILES})
target_link_libraries(TotalCurvature PUBLIC igl_copyleft::cgal polyscope PRIVATE OpenMP::OpenMP_CXX)
```

which version of CMake

Project name

Cmake path

add libraries

Add your project file

Link to libraries

Save this CMakeLists.txt somewhere, and use this as a template to adapt from!

Demos for Libigl-style C++ Code

Now, let’s see some demos!

Take-aways from Today's Lecture

- You learned about the routine of C++ codebase in state of the art graphics research
- You learned about what's CMake
- You saw demos of Libigl-style source code
- You succeeded in checking visual data with Python
- You just got your hands on Libigl and Polyscope, two of the most popular libraries in the research world of computer graphics

Now, your turn!

Go to the course webpage to download data!

We’ll wok on visualizing these data together!

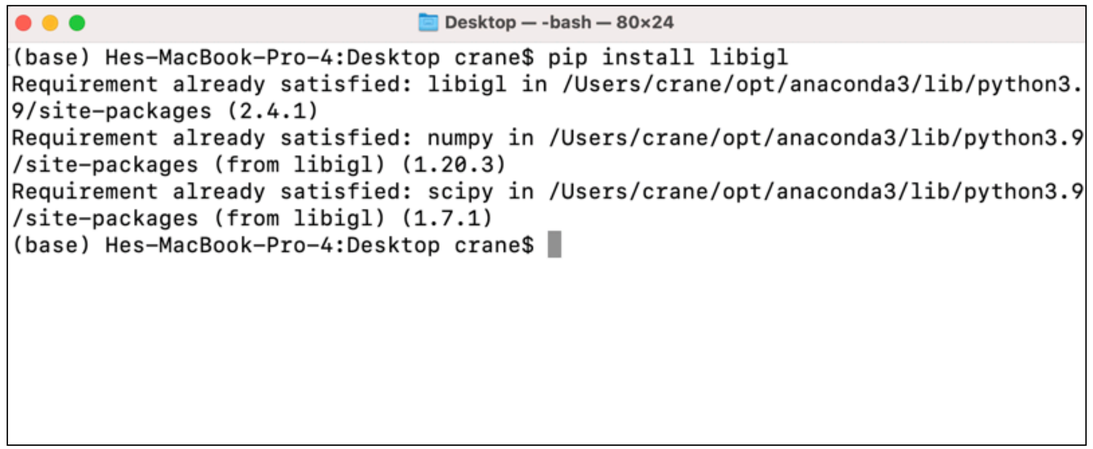
Resolving a Common Confusion from Last Week

Terminal/CommandPrompt and IDE
are not the same!

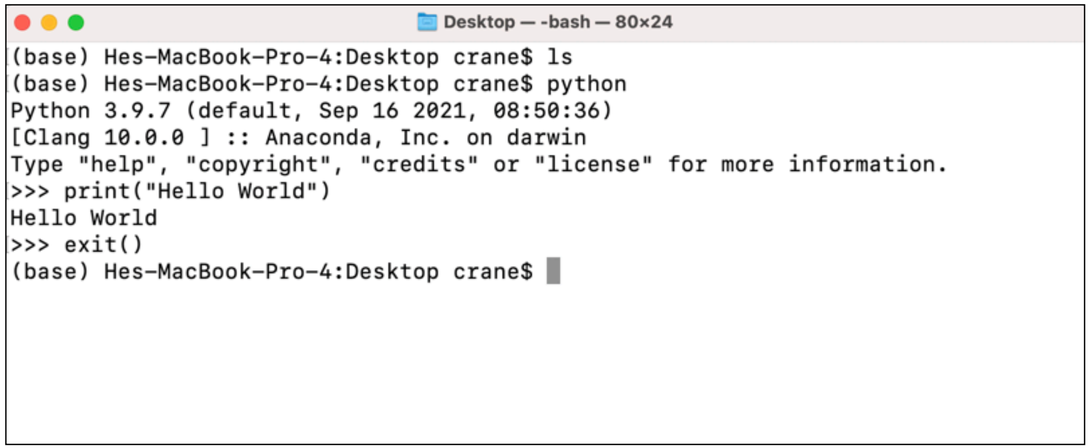
Resolving a Common Confusion from Last Week

Terminal/CommandPrompt

- you are talking to the operating system
- usually use it to install libraries
- you can use it to run simple python code



1. Tell the computer, you want to start writing python
2. Tell the computer, you want to exit python writing and go back to communicating with the system

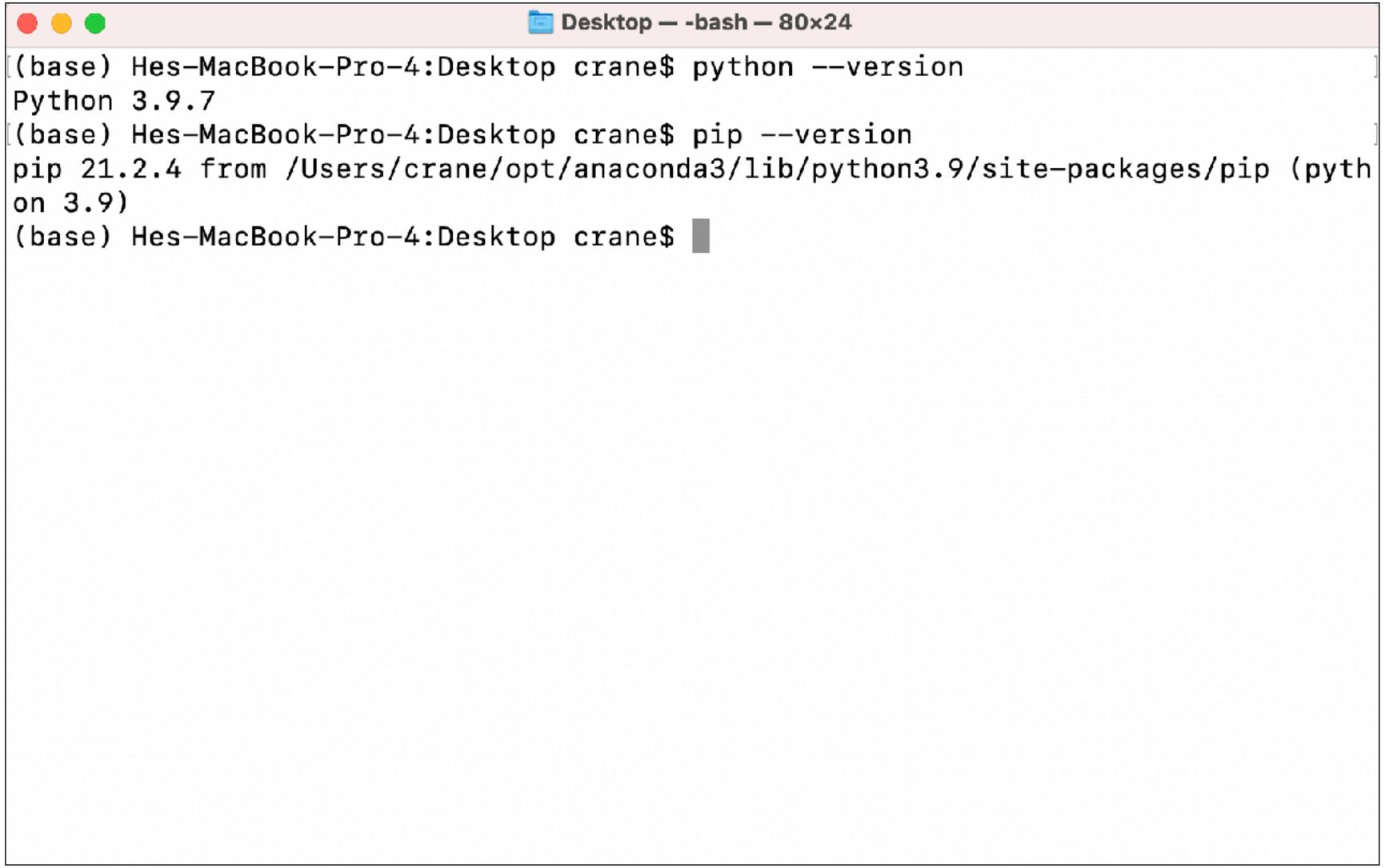


Resolving a Common Confusion from Last Week

IDE (Integrated Develop Environment)

- you are not talking to the operating system
- consider this as a notebook, where you write Python/C++/Java
- this notebook is powerful, it setup an environment where you run your code

Check if you got Python and Pip installed



Pair-Coding

Terminal/CommandPrompt

```
pip install numpy
```

```
pip install Pillow
```

```
pip install libigl
```

```
pip install polyscope
```

Pair-Coding

IDE

```
from PIL import Image

# Open an image file
image_path = "example.jpg"  # Replace with the path to your image file
image = Image.open(image_path)

# Get dimensions
width, height = image.size
print(f"Image Dimensions: Width = {width}, Height = {height}")

# Show the image
image.show()
```

Pair-Coding

IDE

```
import igl  # Import the libigl library
import polyscope as ps  # Import the polyscope library
import numpy as np

# Read the mesh from a file
v, f = igl.read_triangle_mesh("HappyDragon.ply")

# Create a rotation matrix for 90 degrees rotation around x-axis
angle = np.radians(90)
rotation_matrix = np.array([[1, 0, 0],
                            [0, np.cos(angle), -np.sin(angle)],
                            [0, np.sin(angle), np.cos(angle)]])
# Rotate vertices with the matrix
v = np.dot(v, rotation_matrix)

# Print the dimensions of V (vertices) and F (faces)
print("Vertices shape:", v.shape)
print("Faces shape:", f.shape)

# Initialize Polyscope
ps.init()
ps.set_ground_plane_mode("shadow_only")  # set +Z as up direction
ps.set_shadow_darkness(0.1)              # lighter shadows
# Register the mesh in Polyscope
ps_mesh = ps.register_surface_mesh("my_mesh", v, f)
ps_mesh.set_color((68/255,254/255,157/255))
ps_mesh.set_edge_color((0.36,0.36,0.36))  # white edges
ps_mesh.set_edge_width(1.5)      # adjust as needed
# Show the Polyscope GUI
ps.show()
```

Are There Any Questions?