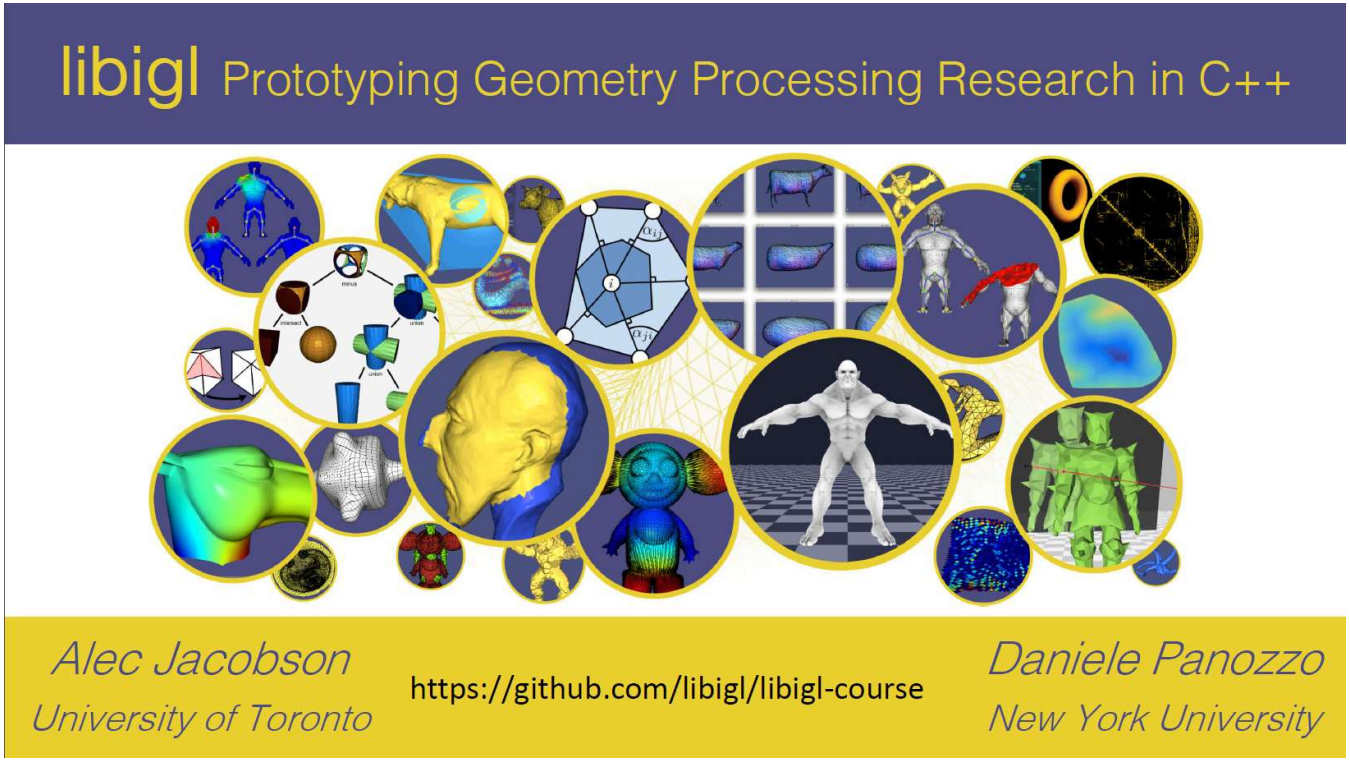
ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εισαγωγή στα Γραφικά Υπολογιστών

Εργαστήριο 8

Εισαγωγή στην Libigl



Εισαγωγή στην Libigl

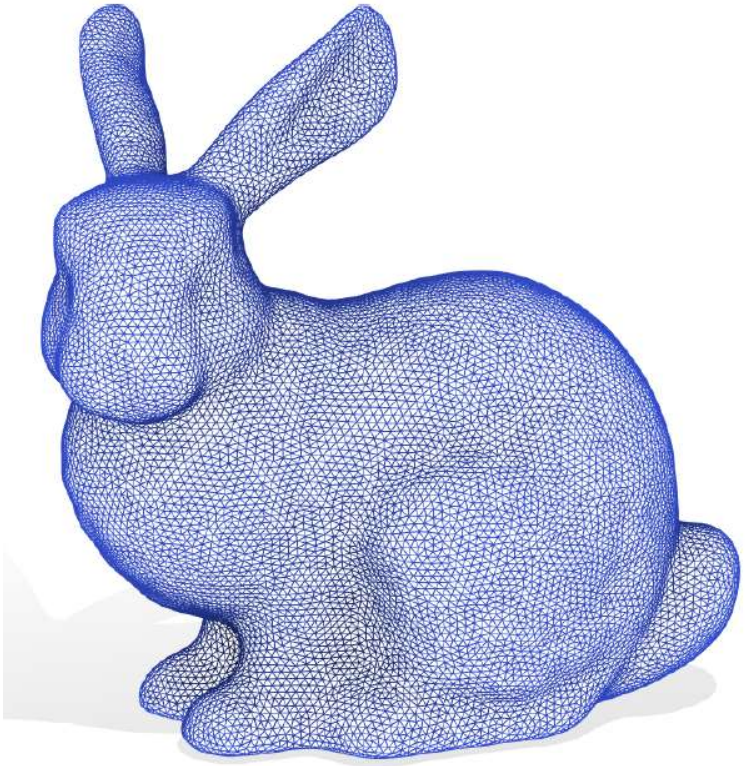
- Η libigl είναι μία βιβλιοθήκη εκτενούς μεγέθους, η οποία περιέχει ποικίλες υλοποιήσεις συναρτήσεων και αλγορίθμων για την επεξεργασία γεωμετρικής πληροφορίας.
- Πρόκειται για μία ‘header only library’, δηλαδή για κάθε συνάρτηση περιέχονται .cpp και .h αρχεία χωρίς υπάρχει κάποια βιβλιοθήκη (.lib αρχείο).
- Η γλώσσα υλοποίησης είναι η C++ σε συνδυασμό με την ‘Eigen’, ακόμα μία ‘header only library’, η οποία καθιστά δυνατή την δημιουργία πινάκων και διανυσμάτων καθώς και πράξεις γραμμικής άλγεβρας μεταξύ αυτών.

Εισαγωγή στην Libigl

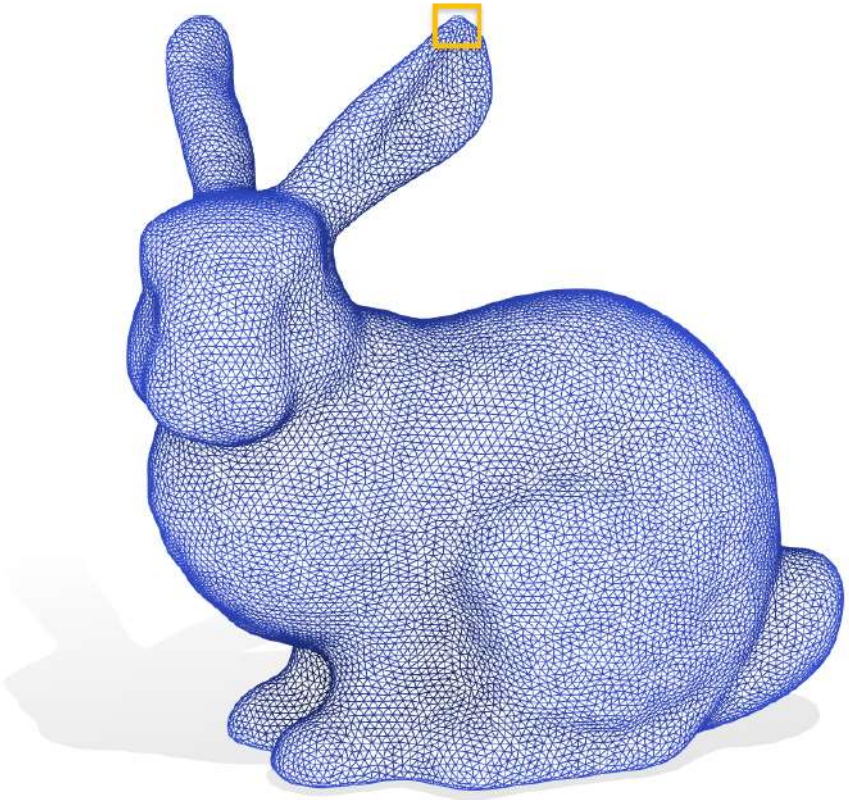
- Μερικές δυνατότητες της Libigl είναι :

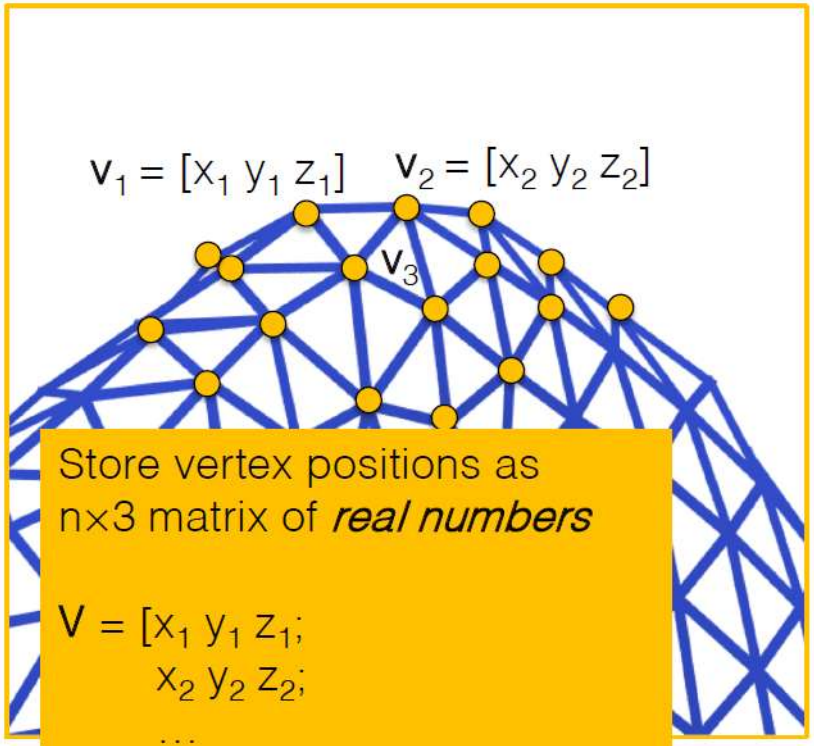
  - Το διάβασμα και η εγγραφή ποικίλων αρχείων (.obj, .off, .stl, .ply, .wrl, .mesh) με τα πιο συνηθισμένα να είναι τα .obj και .off.
  - Η εύρεση επιφάνειας των προσώπων, κανονικών διανυσμάτων, γωνιών μεταξύ κορυφών, μήκους ακμών και γειτνίασης μεταξύ κορυφών.
  - Ο υπολογισμός της Λαπλασιανής, της Βαθμίδας και της Απόκλισης πάνω στην επιφάνια ενός αντικειμένου.
  - Η εξομάλυνση, η παραμόρφωση, η μείωση και η αύξηση της γεωμετρίας ενός αντικειμένου.
  - Η απόδοση χαρακτηριστικών της γεωμετρίας με τη βοήθεια μετρικών.

Meshes



Meshes

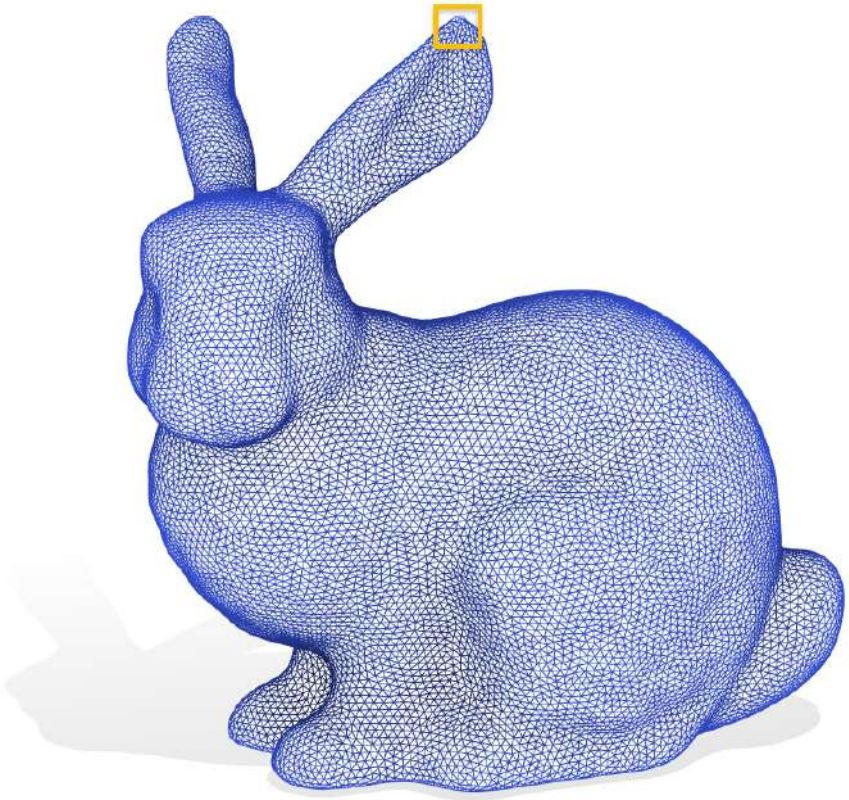


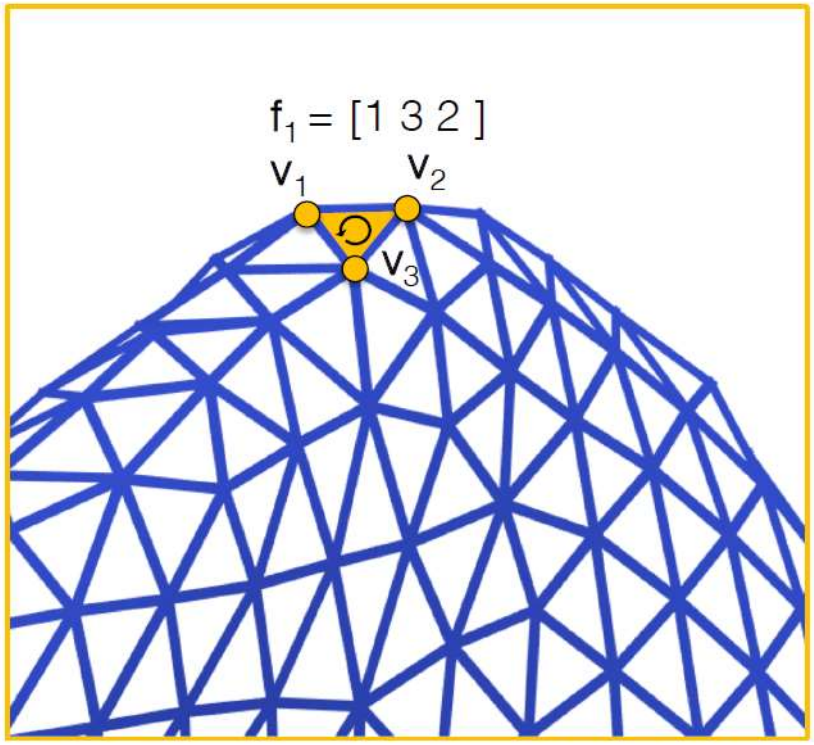


Store vertex positions as
n×3 matrix of ***real numbers***

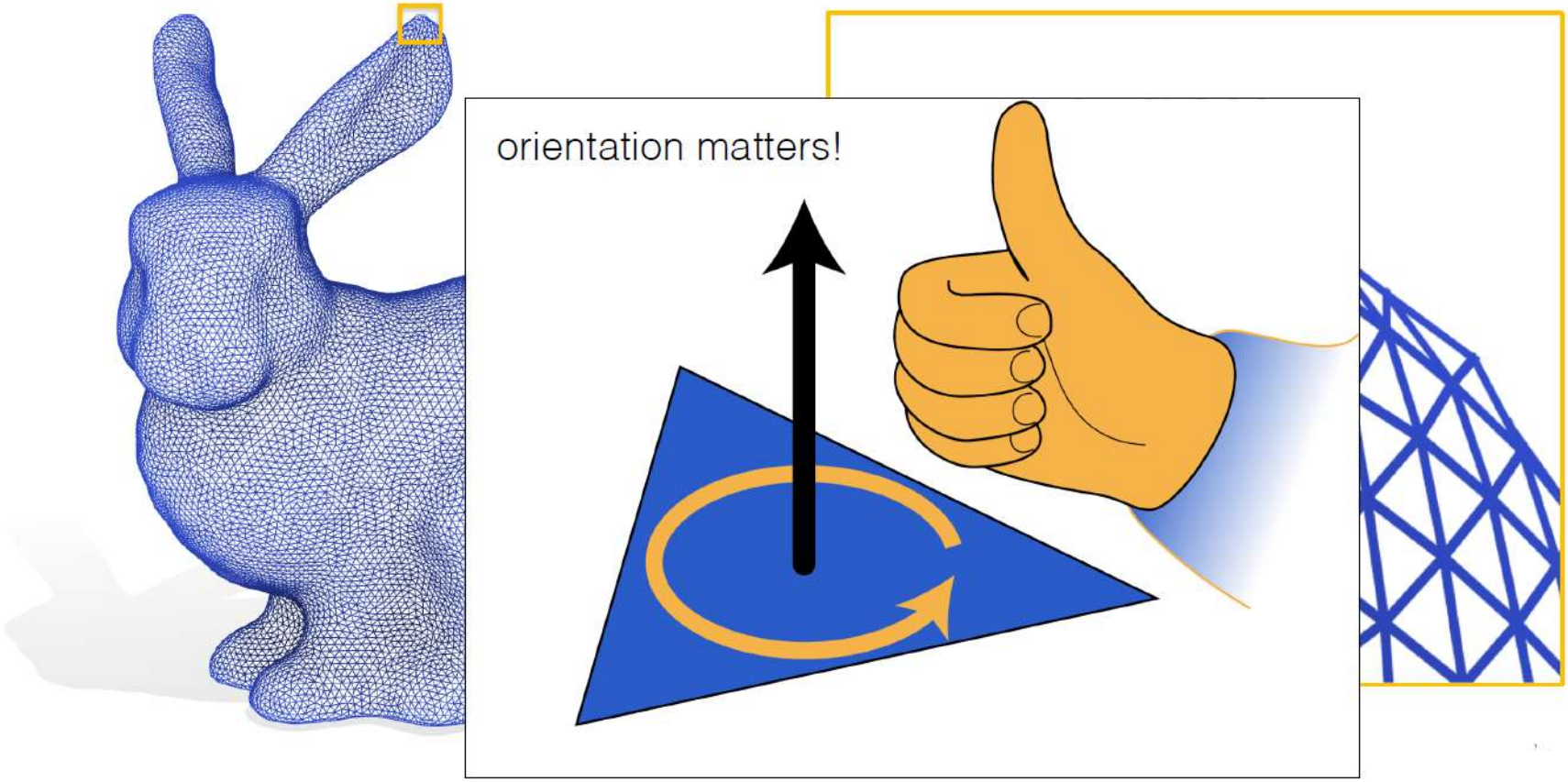
$$V = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \dots & & \\ x_n & y_n & z_n \end{bmatrix}$$

Meshes

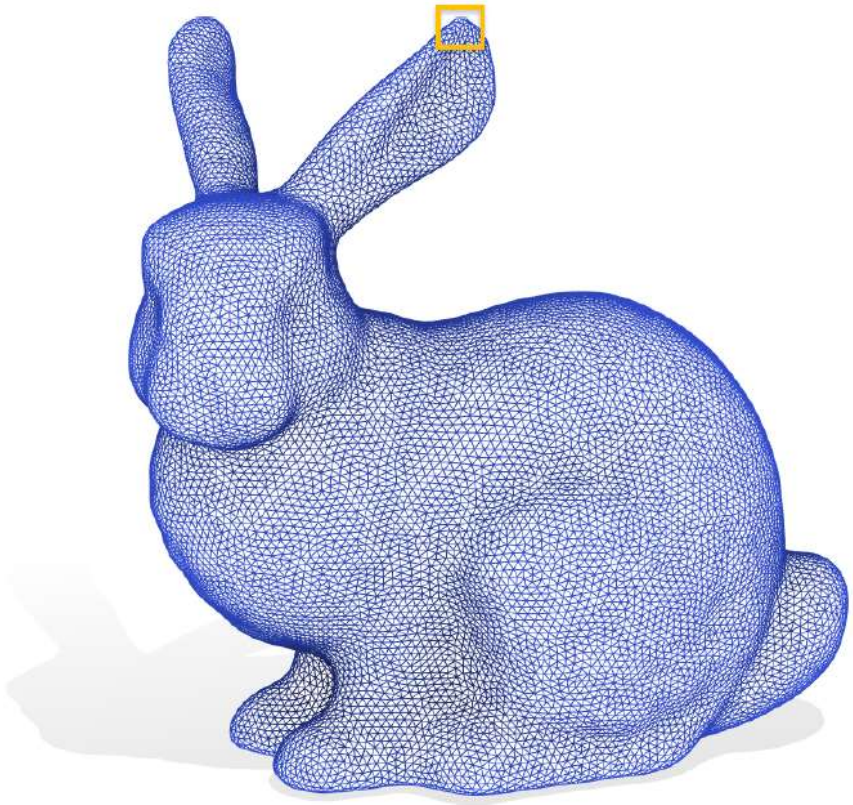


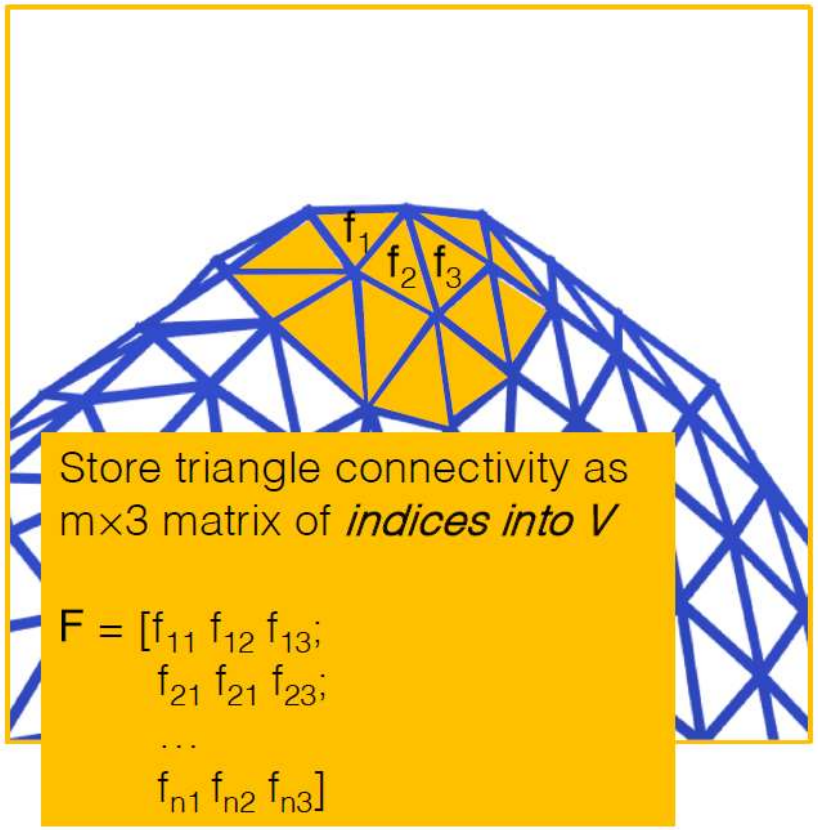


Meshes



Meshes




f_1
f_2
f_3
Store triangle connectivity as m×3 matrix of *indices into V*
F = [f_{11} f_{12} f_{13};
f_{21} f_{21} f_{23};
...
f_{n1} f_{n2} f_{n3}]

File I/O

- Δήλωση πινάκων :

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

Or

  - *Eigen::MatrixXd V;*
  - *Eigen::MatrixXi F;*

File I/O

- Διάβασμα και εγγραφή αρχείου :
  - Για κάθε συνάρτηση πρέπει να γίνεται το ανάλογο ‘include’
  - *#include <igl/readOFF.h>*
  - *#include <igl/writeOBJ.h>*
  - Οι εντολές read και write με τον τύπο του αρχείου (OFF, OBJ, TFG, STL, MSH, MESH).
  - ***readOFF(“path”, V, F)***
  - ***writeOBJ(“path”, V, F)***

File I/O

- Αρχείο OFF :

1. OFF
2. 4 2 5
3. 0.0 0.0 0.0
4. 1.0 0.0 0.0
5. 1.0 1.0 1.0
6. 2.0 1.0 0.0
7. 3 0 1 2
8. 3 1 3 2

1. Στην πρώτη γραμμή ο τύπος του αρχείου (προαιρετικό).
2. Στην δεύτερη γραμμή ο αριθμός των κορυφών, μετά ο αριθμός των προσώπων και τέλος ο αριθμός των ακμών (ο αριθμός των ακμών μπορεί να παραληφθεί γράφοντας 0).
3. Στη συνέχεια οι συντεταγμένες (x, y, z) όλων των κορυφών.
4. Τέλος ο αριθμός των κορυφών κάθε προσώπου και οι δείκτες στις γραμμές του πίνακα με τις κορυφές.

File I/O

- Αρχείο OBJ :

1. Λίστα με τις x,y,z συντεταγμένες των κορυφών **v**.
2. Λίστα με τις συντεταγμένες του texture **vt** (προερετικό).
3. Λίστα με τα κανονικά διανύσματα των κορυφών **vn**. (προερετικό).
4. Λίστα με δείκτες για κάθε πρόσωπο f : δείκτης_κορυφής/δείκτης_texture/δείκτης_κανονικού_διανύσματος.

```
# This file uses centimeters as

v -0.500000 -0.500000 0.500000
v 0.500000 -0.500000 0.500000
v -0.500000 0.500000 0.500000
v 0.500000 0.500000 0.500000
v -0.500000 0.500000 -0.500000
v 0.500000 0.500000 -0.500000
v -0.500000 -0.500000 -0.500000
v 0.500000 -0.500000 -0.500000
vt 0.375000 0.000000
vt 0.625000 0.000000
vt 0.375000 0.250000
vt 0.625000 0.250000
vt 0.375000 0.500000
vt 0.625000 0.500000
vt 0.375000 0.750000
vt 0.625000 0.750000
vt 0.375000 1.000000
vt 0.625000 1.000000
vt 0.875000 0.000000
vt 0.875000 0.250000
vt 0.125000 0.000000
vt 0.125000 0.250000
f 1/1 2/2 3/3
f 3/3 2/2 4/4
f 3/3 4/4 5/5
f 5/5 4/4 6/6
f 5/5 6/6 7/7
f 7/7 6/6 8/8
f 7/7 8/8 1/9
f 1/9 8/8 2/10
f 2/2 8/11 4/4
f 4/4 8/11 6/12
f 7/13 1/1 5/14
f 5/14 1/1 3/3
```

Draw Mesh

- Απεικόνιση αντικειμένου :
  - *#include igl/opengl/glfw/Viewer.h*
  - Ορισμός της μεταβλητής του παραθύρου.
  - ***igl::opengl::glfw::Viewer viewer***
  - Απόδωση δεδομένων στο παράθυρο (η set_mesh αντιγράφει το αντικείμενο στο παράθυρο).
  - ***viewer.data().set_mesh(V, F)***
  - Δημιουργία του παραθύρου, ενός OpenGL context και έναρξη του draw loop.
  - ***viewer.launch()***

Events

- Γενικά μπορεί να ειπωθεί ότι τα events είναι δράσεις που καταλαβαίνει το λογισμικό όπως κίνηση του ποντικιού ή πάτημα κάποιου πλήκτρου.

- Έτσι τα events πυροδοτούν τις callback συναρτήσεις οι οποίες εκτελούν μία διαδικασία που μπορεί να ορίσει ο χρήστης.

- Τα διαθέσιμα callbacks φαίνονται παρακάτω.

```
bool (*callback_pre_draw)(Viewer& viewer);
bool (*callback_post_draw)(Viewer& viewer);
bool (*callback_mouse_down)(Viewer& viewer, int button, int modifier);
bool (*callback_mouse_up)(Viewer& viewer, int button, int modifier);
bool (*callback_mouse_move)(Viewer& viewer, int mouse_x, int mouse_y);
bool (*callback_mouse_scroll)(Viewer& viewer, float delta_y);
bool (*callback_key_down)(Viewer& viewer, unsigned char key, int modifiers);
bool (*callback_key_up)(Viewer& viewer, unsigned char key, int modifiers);
```

Events

- Για την αξιοποίηση των events πρέπει να δημιουργηθεί μία συνάρτηση η οποία ικανοποιεί κατά γράμμα τα ορίσματα της callback συνάρτησης που είναι διαθέσιμη από την libigl.
- Για παράδειγμα στην περίπτωση του πατήματος ενός πλήκτρου πρέπει να αποδοθεί μία συνάρτηση του χρήστη στην αντίστοιχη callback με την εντολή :
  - ***viewer.callback_key_down = &UserFunctionKeyDown()***

Events

- Για την αλλαγή του αντικειμένου πρέπει πρώτα να διαγραφούν τα δεδομένα του παραθύρου.
    - ***viewer.data().clear();***
- Στην συνέχεια αποδίδεται στο παράθυρο το νέο αντικείμενο.
    - ***viewer.data().mesh_mesh(V, F);***
- Τέλος η κάμερα προσαρμόζεται ώστε να φαίνεται όλο το αντικείμενο.
    - ***viewer.core().align_camera_center(V, F);***

Colors

- Για την απόδοση χρώματος σε ένα αντικείμενο μπορεί να χρησιμοποιηθεί η εντολή

  - ***viewer.data().set)colors(C);***

- Ο πίνακας C πρέπει να έχει μέγεθος nx3 (n ο αριθμός των κορυφών) ή mx3 (m ο αριθμός των προσώπων). Εφόσον η πληροφορία για το χρώμα έχει 3 χρωματικά κανάλια rgb (red, green, blue) πρέπει ο πίνακας να έχει 3 στήλες και οι τιμές των χρωμάτων πρέπει να είναι από 0-1.

Colors

- Στο παράδειγμα πραγματοποιείται κανονικοποίηση των διανυσμάτων θέσης σε 0 – 1.
- Αυτό προκύπτει αφαιρώντας από κάθε συντεταγμένη (x, y, z) το ελάχιστο για την αντίστοιχη συντεταγμένη και στη συνέχεια διαιρώντας με το μέγιστο μείον το ελάχιστο για την αντίστοιχη συντεταγμένη .
- Συνεπώς ανάλογα με την θέση της κορυφής αποδίδεται διαφορετικό χρώμα.

- Για πράξεις πινάκων με Arrays και Matrices επισκεφτείτε τους συνδέσμους :
  - https://eigen.tuxfamily.org/dox/group__TutorialArrayClass.html
  - https://eigen.tuxfamily.org/dox/group__TutorialMatrixClass.html

Overlays

- Η Libigl υποστηρίζει την οπτικοποίηση σημείων, γραμμών και επιτικετών στο παράθυρο εμφάνισης.
- Για την οπτικοποίηση ενός σημείου χρησιμοποιείται η εντολή : ***viewer.data().add_points*** η οποία σχεδιάζει ένα σημείο με χρώμα r,g,b για κάθε γραμμή του πίνακα P ο οποίος πρέπει να έχει διαστάσεις Px3.
  - ***viewer.data().add_points(P, Eigen::RowVector3d(r, g, b));***
- Επίσης με την εντολή *viewer.data().point_size = pointSize* μπορεί να οριστεί το μέγεθος του σημείου.

Overlays

- Για την οπτικοποίηση μίας γραμμής χρησιμοποιείται η εντολή : ***viewer.data().add_edges*** η οποία σχεδιάζει μία γραμμή με χρώμα r, g, b για κάθε γραμμή των πινάκων P1 και P2 οι οποίοι πρέπει να έχουν διαστάσεις Px3.

  - ***viewer.data().add_edges(P1, P2, Eigen::RowVector3d(r, g, b));***

Overlays

- Για την δημιουργεία μίας ετικέτας χρησιμοποιείται η εντολή : ***viewer.data().add_label*** η οποία σχεδιάζει μία ετικέτα που περιέχει το αλφαριθμιτικό str στο σημείο p το οποίο είναι ένα διάνυσμα με 3 συντεταγμένες (x, y, z).

  - ***viewer.data().add_label(p, str);***

- Επίσης πρέπει να ενεργοποιηθεί η δυνατότητα εμφάνισης της ετικέτας με την εντολή ***viewer.data().show_custom_labels = true***

- Για αναπαράσταση της ετικέτας χρησιμοποιείται η ImGui (στη συνέχεια θα αναλυθεί) ηοποία καθιστά δυνατή την εμφάνιση της.

Viewer Menu

- Με τη βοήθεια της ImGui υπάρχει η δυνατότητα να δημιουργηθεί ένα μενού μέσω του οποίου να επιτρέπεται η φόρτωση νέων μοντέλων, μεταβλητών που επηρεάζουν την οπτική εμφάνιση και την γεωμετρική πληροφορία των αντικειμένων κ.τ.λ.π.
- Για την δημιουργεία ενός μενού θα πρέπει να προστεθούν τα παρακάτω includes.

```
#include <igl/opengl/glfw/imgui/ImGuiMenu.h>
#include <igl/opengl/glfw/imgui/ImGuiHelpers.h>
#include <imgui/imgui.h>
```

Viewer Menu

- Αρχικά όπως και με τον viewer πρέπει να αρχικοποιηθεί ένα μενού με την εντολή:
  - ***igl::opengl::glfw::imgui::ImGuiMenu menu;***
- Στη συνέχεια πρέπει να προστεθεί στον viewer το μενού που μόλις δημιουργήθηκε με την εντολή:
  - ***viewer.plugins.push_back(&menu);***

Viewer Menu

- Υπάρχει η δυνατότητα να χρησιμοποιηθεί ένα default παράθυρο το οποίο περιέχει πολλά έτοιμα εργαλεία, χρησιμοποιώντας μία από τις callback συναρτήσεις του μενού η οποία είναι η :

  - ***menu.callback_draw_viewer_menu***

- Η συνάρτηση αυτή επιστέφει void, δεν έχει παραμέτρους και θα πρέπει να περιέχεται μέσα σε αυτήν η εντολή ***menu.draw_viewer_menu()*** *έτσι ώστε να περιέχονται τα έτοιμα εργαλεία.*

Viewer Menu

- Επίσης είναι δυνατή η δημιουργεία ενός νέου παραθύρου χρησιμοποιώντας την callback συνάρτηση : ***menu.callback_draw_custom_window,*** η οποία επιστρέφει void και δεν δέχεται παραμέτρους.

- Σε αυτήν την περίπτωση θα χρειαστεί να οριστεί η θέση, το μέγεθος και το όνομα του παραθύρου.

Multiple Meshes

- Ο Viewer της libigl μπορεί να φορτώσει πολλά αντικείμενα και να φαίνονται όλα ταυτόχρονα.
- Η συνάρτηση ***viewer.load_mesh_from_file(path)*** μπορεί να χρησιμοποιηθεί όταν δεν είναι γνωστός ο τύπος του αρχείου και εφόσον ανήκει στην κλάση *Viewer* προσθέτει κατευθείαν το αντικείμενο στο παράθυρο χωρίς να χρειαστεί η εντολή ***viewer.data().set_mesh(V, F);***
- Κάθε αντικείμενο έχει χαρακτηριστηκό id ξεκινώντας από το 0 για το πρώτο αντικείμενο. Κάθε φορά που προστίθεται ένα καινούργιο αντικείμενο στον viewer το id αυξάνεται κατά 1 και είναι διαθέσιμο με την εντολή **viewer.data().id** ή ***viewer.selected_data_index*** (επιστρέφει το id του τελευταίου αντικειμένου που προστέθηκε).

Multiple Meshes

- Με την συνάρτηση ***viewer.erase_mesh(viewer.selected_data_index)*** διαγράφεται το αντικείμενο με το συγκεκριμένο id.
- Η ***viewer.data_list*** επιστρέφει μία λίστα με όλα τα αντικείμενα του viewer.
- Η callback συνάρτηση ***viewer.callback_pre_draw*** καλείται πριν σχεδιαστεί κάτι στην οθόνη και ανά τακτά χρονικά διαστήματα κατά την εκτέλεση της εφαρμογής. Στο παράδειγμα η δουλεία της είναι να χρωματίζει όλα τα αντικείμενα με διαφορετικό χρώμα εκτός αυτό που είναι επιλεγμένο από τον viewer το οποίο ζωγραφίζετε με κόκκινο χρώμα.

Multiple Views

- Μέχρι στιγμής για την οπτικοποίηση των αντικειμένων χρεισιμοποιείται όλη η οθόνη. Ο viewer μπορεί να χωριστεί σε πυρήνες όπου ο καθένας να έχει το δικό του viewport.
- Για την προσθήκη ενός νέου πυρήνα χρησιμοποιείται η εντολή **viewer.append_core(Eigen::Vector4f(startx, starty, endx, endy))** η οποία επιστρέφει το id του πυρήνα. Ο μέγιστος αριθμός των πυρήνων στον viewer είναι 31.
- Κάθε πυρήνας (όπως και τα αντικείμενα) έχει το μοναδικό αναγνωριστικό του (id) το οποίο λαμβάνεται με την εντολή ***viewer.core_list[index].id***
- Το viewport του πυρήνα μπορεί να οριστεί με την εντολή **viewer.core(coreId).viewport** ή **viewer.core().viewport** αν υπάρχει μόνο ένας πυρήνας.

Multiple Views

- Η callback συνάρτηση ***viewer.callback_init*** δέχεται έναν viewer και επιστρέφει μία boolean. *Χ*ρησιμοποιείται για αρχικοποίηση.
- Η callback συνάρτηση ***viewer.callback_post_resize*** δέχεται έναν viewer, δύο ακεραίους (width , height) και επιστρέφει μία boolean. Καλείται κάθε φορά που αλλάζει μέγεθος το παράθυρο και μία φορά στην αρχή.
- Επειδή κάθε φορά που δημιουργείται ένας πυρήνας όλα τα αντικείμενα του viewer εμφανίζονται σε όλα τα viewport που έχουν δημιουργηθεί, υπάρχει η δυνατότητα να μην είναι ορατό ένα αντικείμενο (mesh_id) σε κάποιον πυρήνα (core_id) με την εντολή ***viewer.data(mesh_id).set_visible(false, core_id)***

Σύνδεσμοι

- https://libigl.github.io/
- https://libigl.github.io/tutorial/
- https://www.youtube.com/watch?v=hsYRSUW-FGA