Its all about 3D!

Alessandro Fasse
OneFID GmbH

28. November 2018

Outline

Motivation

- Real objects around us are all 3D

Motivation

- Real objects around us are all 3D
- So every 1D or 2D data of an object is only a model or simplification

Motivation

- Real objects around us are all 3D
- So every 1D or 2D data of an object is only a model or simplification
- For example taking circumferences at feet is a quite drastic simplification

Motivation

- Real objects around us are all 3D
- So every 1D or 2D data of an object is only a model or simplification
- For example taking circumferences at feet is a quite drastic simplification
- Recent technologies like 3D scanner, 3D printer and CT scanner allows us to record 3 dimensional data

Motivation

- Real objects around us are all 3D
- So every 1D or 2D data of an object is only a model or simplification
- For example taking circumferences at feet is a quite drastic simplification
- Recent technologies like 3D scanner, 3D printer and CT scanner allows us to record 3 dimensional data
- It allows us to get representations that are accurate up to 1mm

Representation of 3D data - Summary

- Parametrization of a surface in terms of vanishing set of a function, *i.e.* consider

$$\{(x, y, z) \in \mathbb{R}^3 \,|\, x^2 + y^2 + z^2 = 1\}.$$

Representation of 3D data - Summary

- Parametrization of a surface in terms of vanishing set of a function, *i.e.* consider

$$\{(x, y, z) \in \mathbb{R}^3 \,|\, x^2 + y^2 + z^2 = 1\}.$$

- Then we have that the sphere $S^2 = f^{-1}(0)$ where $f(x, y, z) = x^2 + y^2 + z^2 - 1$.

Representation of 3D data - Summary

- Parametrization of a surface in terms of vanishing set of a function, *i.e.* consider

$$\{(x, y, z) \in \mathbb{R}^3 \,|\, x^2 + y^2 + z^2 = 1\}.$$

- Then we have that the sphere $S^2 = f^{-1}(0)$ where $f(x, y, z) = x^2 + y^2 + z^2 - 1$.
- NURBS: Is a representation of a surface via (multi-dimensional) splines and control points

Representation of 3D data - Summary

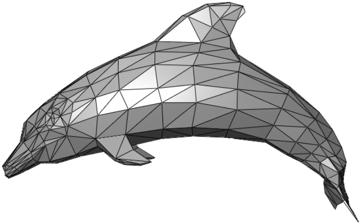
- Parametrization of a surface in terms of vanishing set of a function, *i.e.* consider

$$\{(x, y, z) \in \mathbb{R}^3 \,|\, x^2 + y^2 + z^2 = 1\}.$$

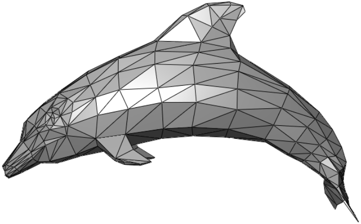
- Then we have that the sphere $S^2 = f^{-1}(0)$ where $f(x, y, z) = x^2 + y^2 + z^2 - 1$.
- NURBS: Is a representation of a surface via (multi-dimensional) splines and control points
- **Discretization:** Dividing a mesh into (planar) simpler subfaces and glue them together to obtain an approximation to the whole mesh.

Representation of 3D data - Triangular meshes

- The principal idea is to represent the surface of a mesh as a set of triangle

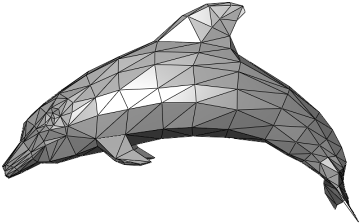


- The principal idea is to represent the surface of a mesh as a set of triangle
- Store the coordinates of vertices in a $n \times 3$ double (or float) list called `vertices`



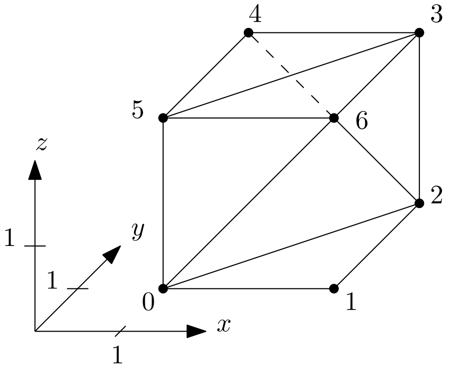
Representation of 3D data - Triangular meshes

- The principal idea is to represent the surface of a mesh as a set of triangle
- Store the coordinates of vertices in a $n \times 3$ double (or float) list called `vertices`
- Store which vertices make up a triangle in a $m \times 3$ unsigned integer list called `faces`



Representation of 3D data - Triangular meshes, Example

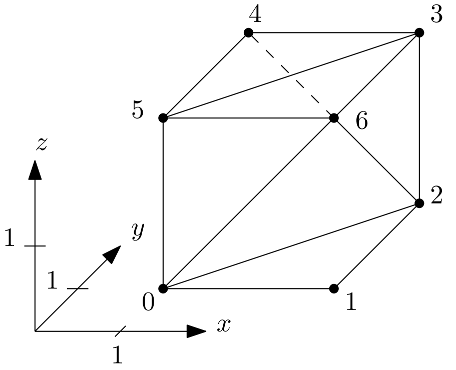
- Consider the following picture with a cube of edge length 2



Representation of 3D data - Triangular meshes, Example

- Consider the following picture with a cube of edge length 2
- `vertices`

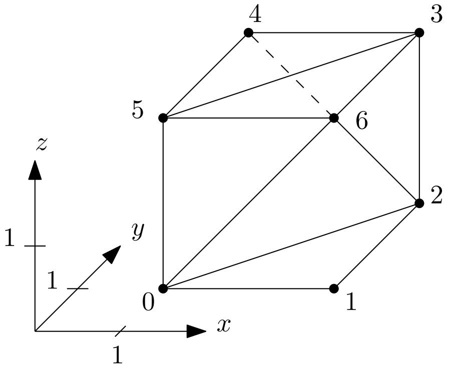
$$\begin{pmatrix} 1 & 1 & 0 \\ 3 & 1 & 0 \\ 3 & 3 & 0 \\ 3 & 3 & 2 \\ 1 & 3 & 2 \\ 1 & 1 & 2 \\ 3 & 1 & 2 \end{pmatrix}$$



Representation of 3D data - Triangular meshes, Example

- Consider the following picture with a cube of edge length 2
- `vertices`

$$\begin{pmatrix} 1 & 1 & 0 \\ 3 & 1 & 0 \\ 3 & 3 & 0 \\ 3 & 3 & 2 \\ 1 & 3 & 2 \\ 1 & 1 & 2 \\ 3 & 1 & 2 \end{pmatrix}$$

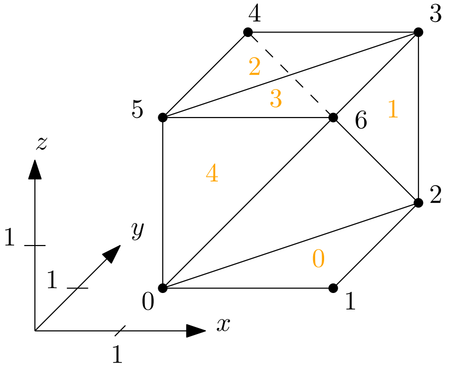


- Note that in general coordinates are real numbers

Representation of 3D data - Triangular meshes, Example

- `faces`

$$\begin{pmatrix} 0 & 2 & 1 \\ 3 & 2 & 4 \\ 4 & 5 & 3 \\ 5 & 6 & 3 \\ 5 & 0 & 6 \end{pmatrix}$$

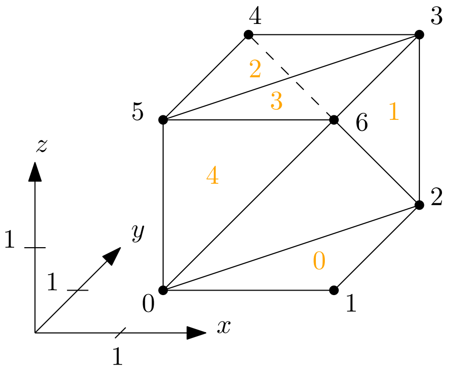


Representation of 3D data - Triangular meshes, Example

- `faces`

$$\begin{pmatrix} 0 & 2 & 1 \\ 3 & 2 & 4 \\ 4 & 5 & 3 \\ 5 & 6 & 3 \\ 5 & 0 & 6 \end{pmatrix}$$

- What about taking $(2\,1\,0)$ instead of $(0\,2\,1)$ for the first face?



Representation of 3D data - Triangular meshes, Normal convention

- Face indices are given such that we can define a normal vector

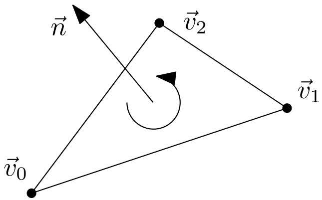
Representation of 3D data - Triangular meshes, Normal convention

- Face indices are given such that we can define a normal vector
- Let k be the k-th triangle and $\vec{v}_i = \texttt{vertices}[\texttt{faces}[k, i], :] \in \mathbb{R}^3$ for $i \in \{0, 1, 2\}$.

Representation of 3D data - Triangular meshes, Normal convention

- Face indices are given such that we can define a normal vector
- Let k be the k-th triangle and $\vec{v}_i = \texttt{vertices}[\texttt{faces}[k,i],:] \in \mathbb{R}^3$ for $i \in \{0,1,2\}$.
- Convention:

$$\vec{n} = (\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0)$$



C++ vs. Python

- In general, Python is way to slow for computational geometry

C++ vs. Python

- In general, Python is way to slow for computational geometry
- Consider the problem of finding a ray intersection with a mesh

C++ vs. Python

- In general, Python is way to slow for computational geometry
- Consider the problem of finding a ray intersection with a mesh
- In my example I took a mesh with 215593 vertices and 431182 faces at a size of 13.4 MB as a `ply` file

C++ vs. Python

- In general, Python is way to slow for computational geometry
- Consider the problem of finding a ray intersection with a mesh
- In my example I took a mesh with 215593 vertices and 431182 faces at a size of 13.4 MB as a `ply` file
- I wrote exactly the same code in Python and C++ and run it on this MacBook Pro 2013, 2.9 GHz Intel i5, 16GB 1867 MHz DDR3, Single Core

C++ vs. Python

- In general, Python is way to slow for computational geometry
- Consider the problem of finding a ray intersection with a mesh
- In my example I took a mesh with 215593 vertices and 431182 faces at a size of 13.4 MB as a `ply` file
- I wrote exactly the same code in Python and C++ and run it on this MacBook Pro 2013, 2.9 GHz Intel i5, 16GB 1867 MHz DDR3, Single Core
- In Python it took ≈ 76.71 s

C++ vs. Python

- In general, Python is way to slow for computational geometry
- Consider the problem of finding a ray intersection with a mesh
- In my example I took a mesh with 215593 vertices and 431182 faces at a size of 13.4 MB as a `ply` file
- I wrote exactly the same code in Python and C++ and run it on this MacBook Pro 2013, 2.9 GHz Intel i5, 16GB 1867 MHz DDR3, Single Core
- In Python it took ≈ 76.71 s
- In C++ it took only ≈ 0.011 s and this code was not optimized!

C++ vs. Python

- In general, Python is way to slow for computational geometry
- Consider the problem of finding a ray intersection with a mesh
- In my example I took a mesh with 215593 vertices and 431182 faces at a size of 13.4 MB as a `ply` file
- I wrote exactly the same code in Python and C++ and run it on this MacBook Pro 2013, 2.9 GHz Intel i5, 16GB 1867 MHz DDR3, Single Core
- In Python it took ≈ 76.71 s
- In C++ it took only ≈ 0.011 s and this code was not optimized!
- This is a factor of ≈ 687.45 . With optimized code I think it is possible to even get factors around 1000.

Current research

- Mesh cleaning and reconstruction in order to repair and enhance existing meshes.

Current research

- Mesh cleaning and reconstruction in order to repair and enhance existing meshes.
- Deformation of meshes

- You can download the files at https://github.com/AlessandroFasse/itsallabout3d

Workshop

- You can download the files at https://github.com/AlessandroFasse/itsallabout3d
- You need to have Python3 installed

Workshop

- You can download the files at https://github.com/AlessandroFasse/itsallabout3d
- You need to have Python3 installed
- The exercise sheet is also within the Git