3D Reconstruction

Outline

- 3D reconstruction from images
- Laser scanner
- Point cloud
- obj file format

3D Reconstruction

- 3D reconstruction from images

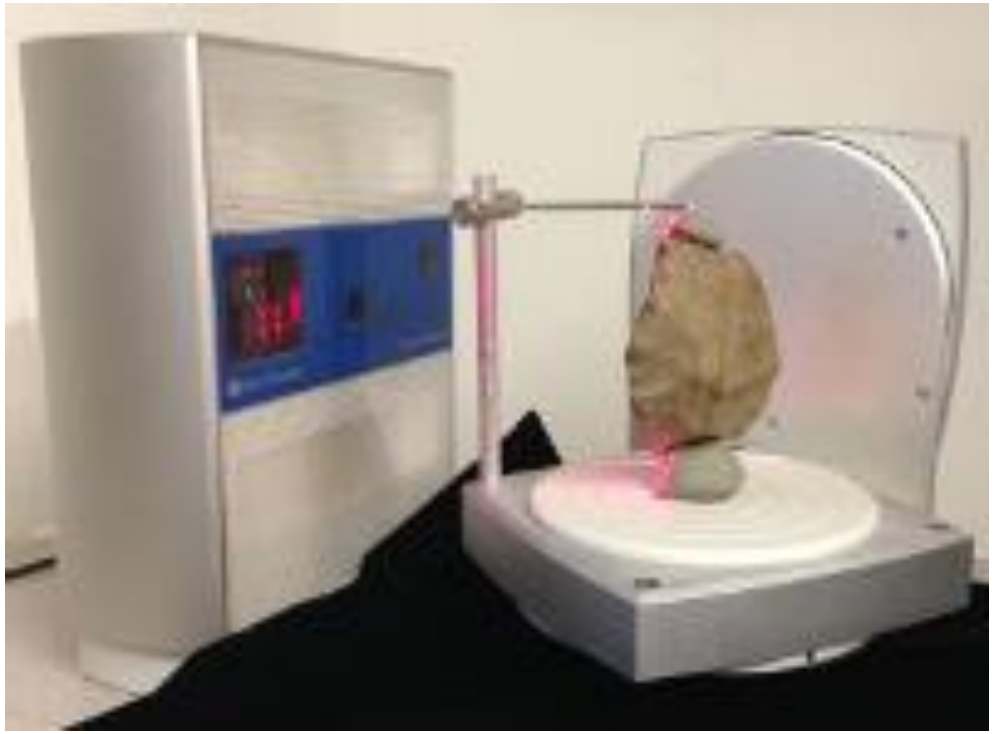
3D Reconstruction

3D laser scanner

What is 3D laser scanning?

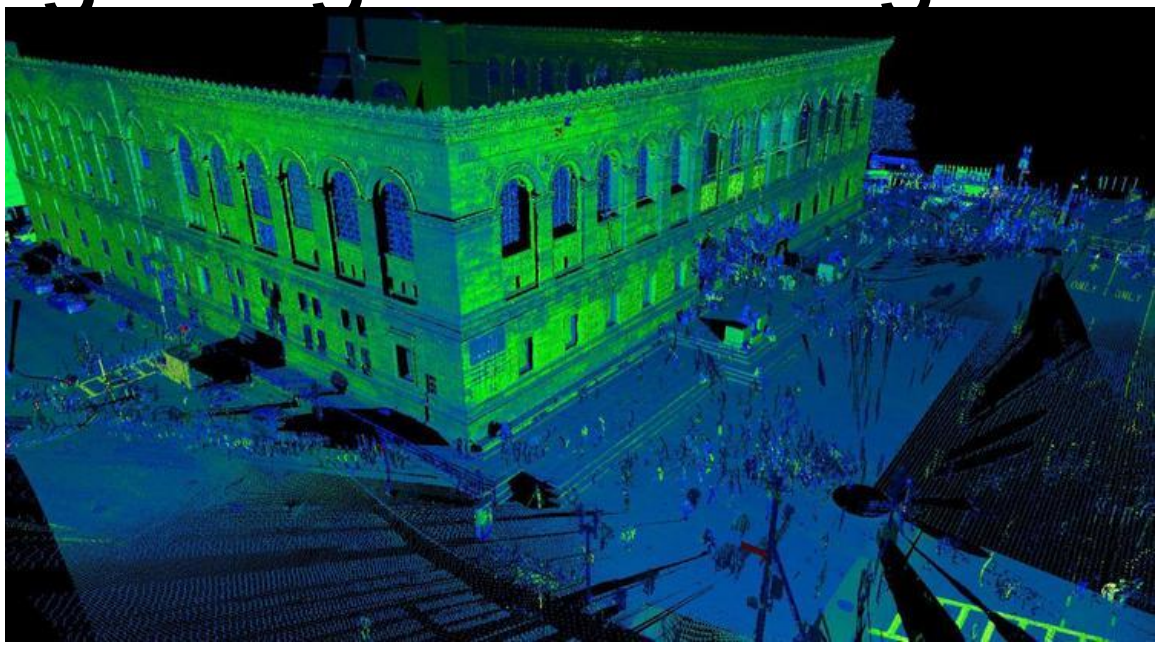
3D Laser Scanning is a technology to capture real scenery in a 3D virtual world.

Short range scanning



Medium range scanning

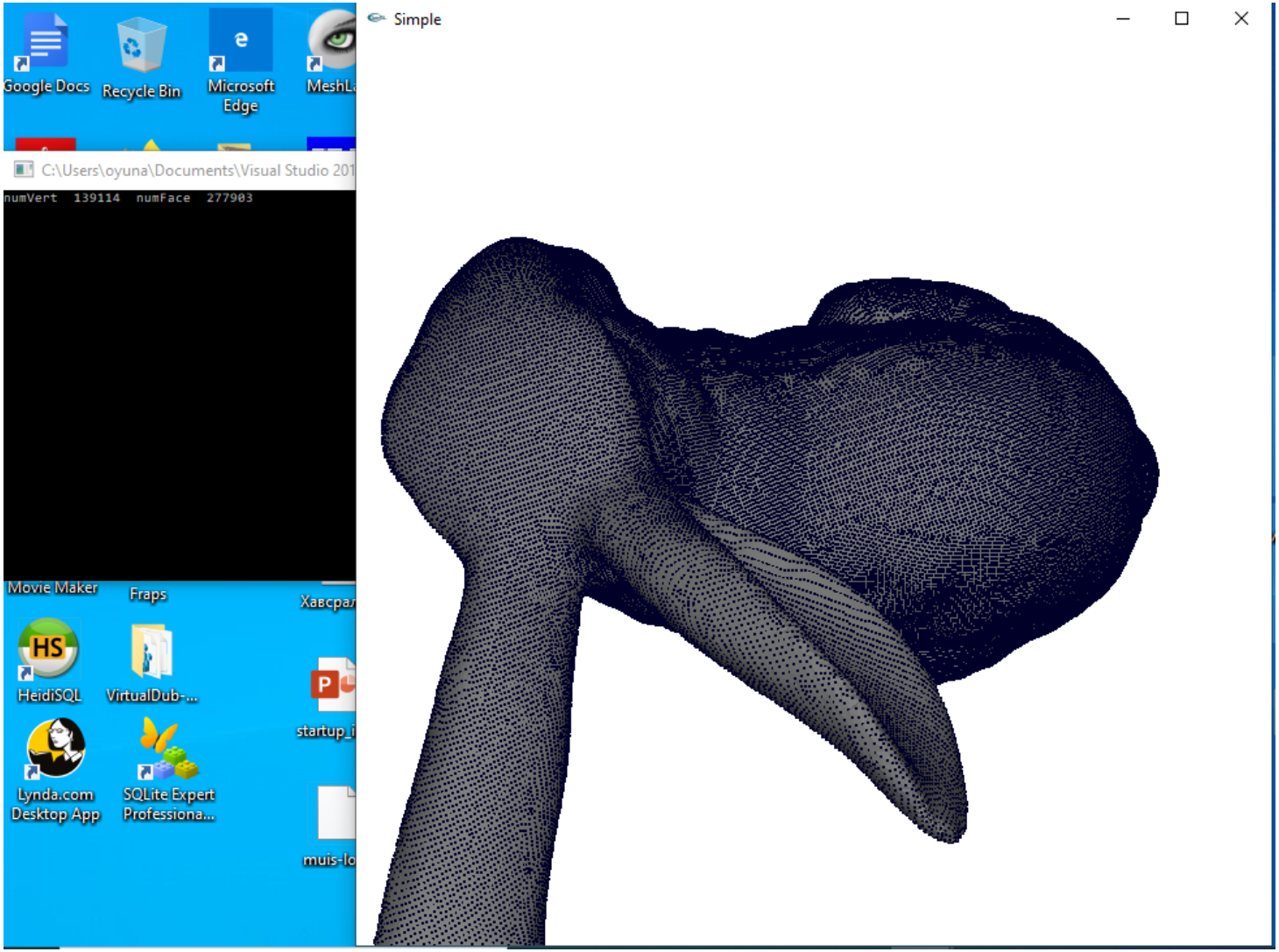
Long range scanning

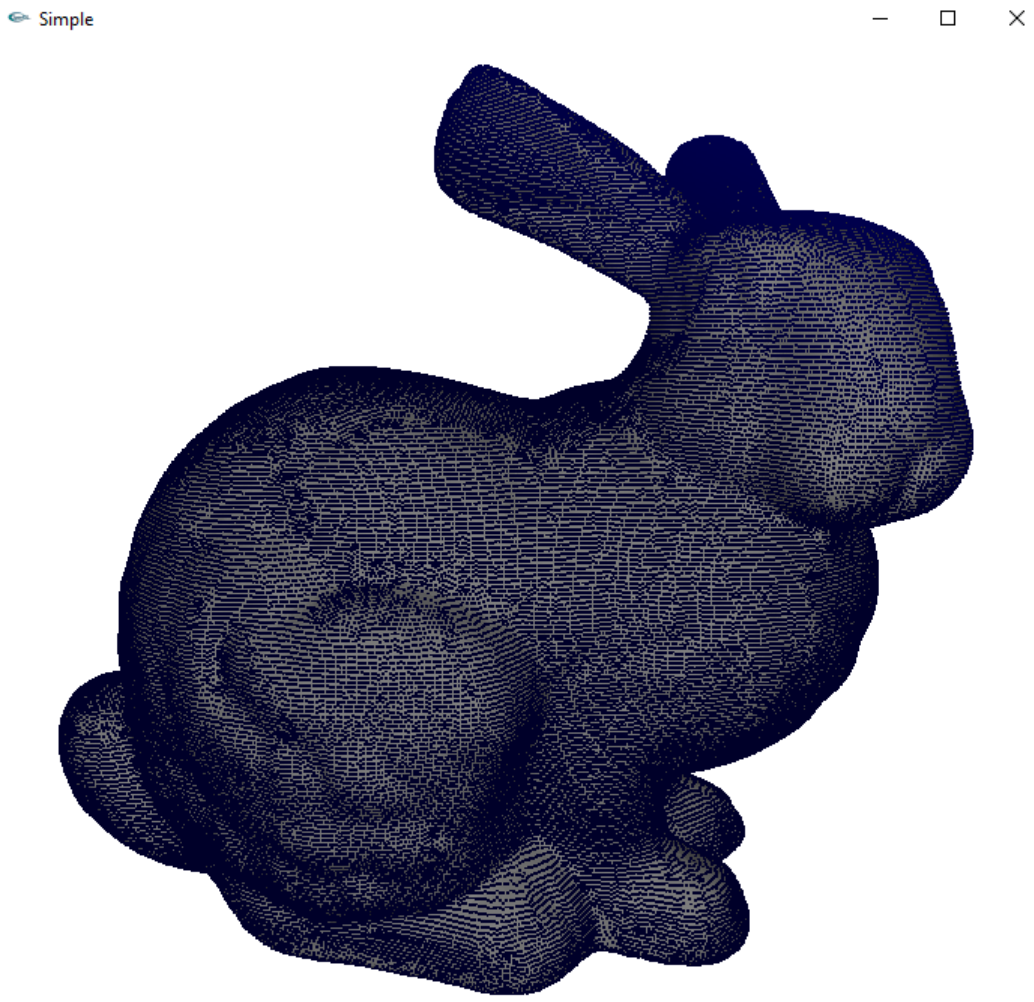


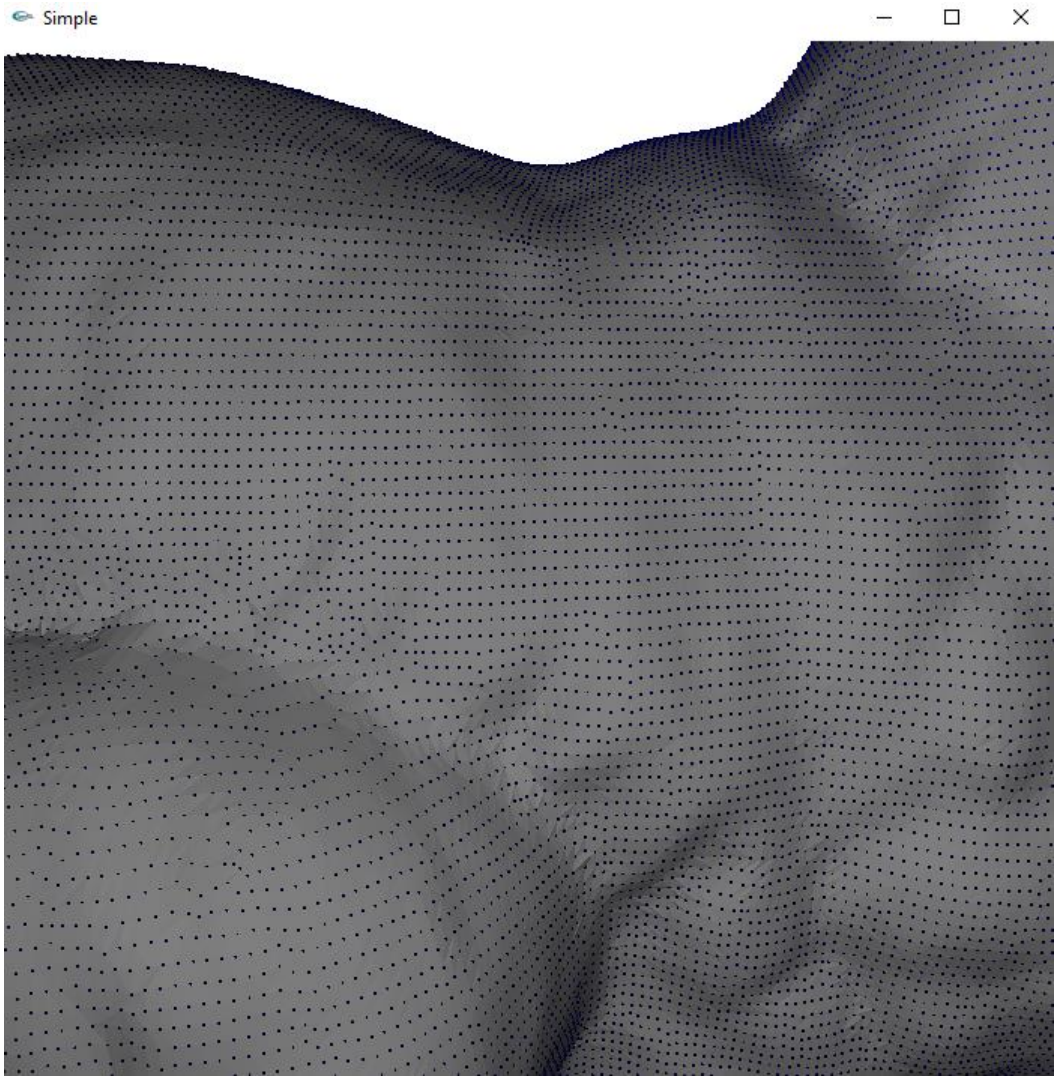
Application of 3D laser scanning

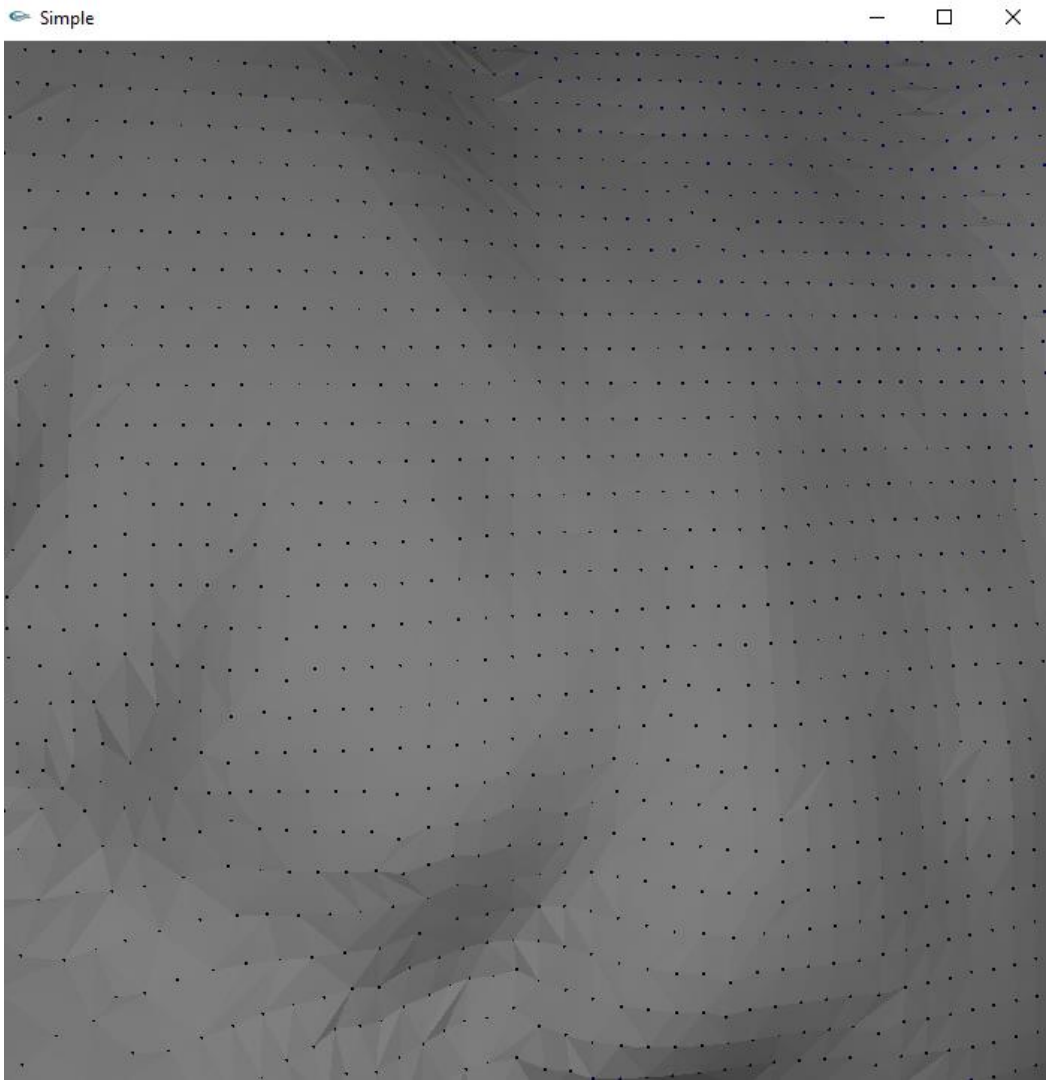
- Industrial
- Architectural
- Civil surveying
- Urban topography
- Mining
- Reverse engineering
- Archaeology
- Dentistry
- Reverse engineering etc.

Point cloud


Google Docs
Recycle Bin
Microsoft Edge
C:\Users\oyuna\Documents\Visual Studio 201
numVert 139114 numFace 277903
Movie Maker
Fraps
HeidiSQL
VirtualDub-...
Lynda.com Desktop App
SQLite Expert Professiona...
Simple


Simple


Simple


Simple

OBJ file format

- Great format for static objects in a game
- Can be exported from Maya, one of the industry standards for creation of 3D content
  - Maya is developed by Alias Wavefront
- Simple, ASCII-based format
- Contains geometry data, no animation

Parsing text files

- Say you have a string, called szVertex, that contains the line:

  Vertex 15 - [14.12, 12.51, 33.10];

- The sscanf function would look like this:

  sscanf(szVertex, "%s %d – [%f, %f, %f];", s, &i, &f1, &f2, &f3);

Parameters for Building a Format String

Parameter	Use
`%s`	A `%s` as a format string means that `scanf` will look for a string. It will read until it finds a null, space, or newline character.
`%c`	`%c` reads a single character. It will read the first character it sees, regardless of what type it is, even a space or a newline.
`%d`	Reading in integers uses `%d`. With integers `scanf` will read until it finds a space, letter, or symbol that is not part of a number. Integers read with `%d` can be positive or negative.
`%f`	`%f` is used for reading real numbers, particularly floating point values. It will cause `scanf` to read a number the same way as `%d`, but it will include decimals.

OBJ file format

- Each line of the model file starts with one or two letters that tell the program what that line is for.
- **v**: A letter v followed by a space is a plain vanilla vertex. If this is the case, following a single space will be three floating-point values, each separated by a single space as well.

OBJ file format

- **vt**: The string vt signifies that the line contains texture coordinates. Each texture coordinate is two floats, again separated by one space.
- **vn**: vn signifies a vertex normal. Other than the prefix, the line mirrors the vertex lines: three floats.

OBJ file format

- f: An f signifies a face. A face is a set of indexes into the arrays of vertices, texcoords, and normals.
- However, only the vertex indexes must be present; the other two are optional. Every vertex index should be positive.

OBJ file format

- Anything else: Any other line of prefixes such as **g** (group), # (comments), **p** (point), **l** (line), **surf** (surface), and **curv** (curve) should be ignored for now.

OBJ file format

- A typical line in the OBJ file that represents a face or triangle and contains only vertices would look something like this:

```
f 1 2 3
```

- This code says that the faces use the first, second, and third vertex indexes to draw the triangles.

OBJ file format

- If the faces are textured, but contain no vertex normals, the syntax would be similar to this:

f 1/4 2/5 3/6

- This line says that the triangles use vertices 1, 2, and 3 and texture indexes 4, 5, and 6.

OBJ file format

- Another variation of this line could use all three types of vertex data, the vertex, texture coordinate, and normal indexes.

f 1/4/7 2/5/8 3/6/9

- The first number is for the vertex itself, the second is for its texture coordinate, and the third is for the vertex normal.

rect - WordPad

File Edit View Insert Format Help

```
# This file uses centimeters as units for non-parametric coordinates.

mtllib rect.mtl
g default
v -2.000000 -0.000000 2.000000
v 2.000000 -0.000000 2.000000
v -2.000000 0.000000 -2.000000
v 2.000000 0.000000 -2.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 0.000000 1.000000
vt 1.000000 1.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
s off
g pPlane1
usemtl initialShadingGroup
f 1/1/1 2/2/2 4/4/3 3/3/4
```

For Help, press F1