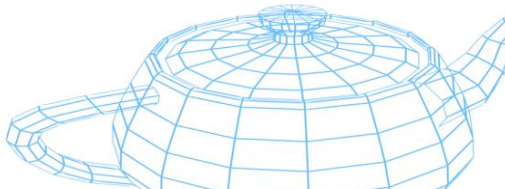
SUPSI

Computer Graphics

3D File Formats

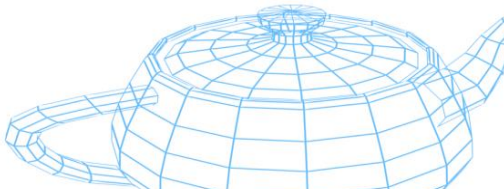
Achille Peternier, adjunct professor



3D file formats

- Most of them were originally native formats used by specific applications:
  - Not designed with inter-operability in mind.
  - Developers started using the same file formats used by their graphics tools and designers.

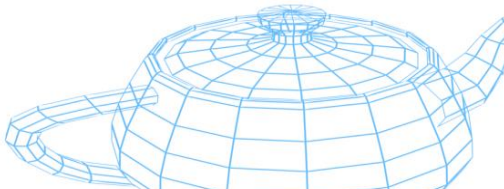
- First significant contributions towards a 3D inter-operable file format definition are relatively recent (e.g., COLLADA in 2004).



3D file formats

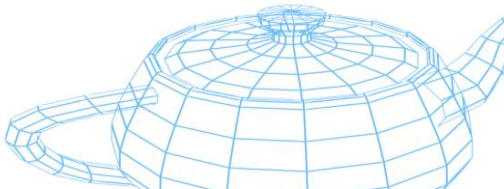
- 3D formats are more like containers of various objects (*à la* PDF) rather than wrappers around a specific set of data (e.g.: image file formats).

- Many 3D formats contain information about the different elements of a scene, like geometric objects, material properties, light sources, etc.:
  - Most 3D file formats contain an entire scene and not only a list of triangles.



OBJ

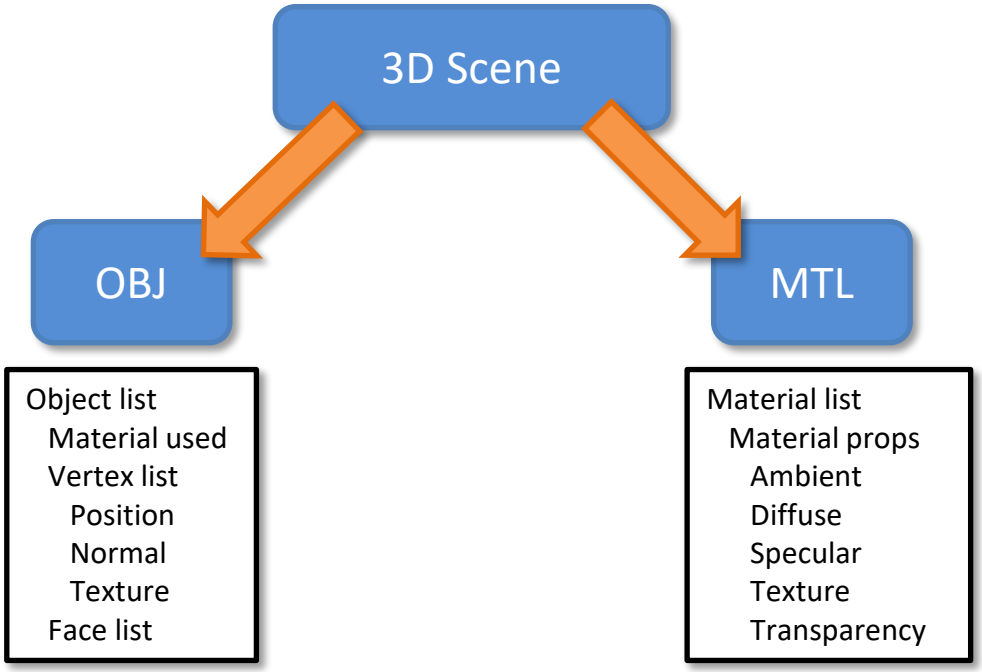
- Introduced by Wavefront Technologies (1980-1990).
- Simple ASCII file format, easy to parse.
- Extension: **.obj** (not to confuse with the files used by C/C++ compilers!)
- Objects are stored in world coordinates:
  - Difficult to apply additional transformations once exported as OBJ.
- This format does not support animation, light sources/cameras, nor hierarchies.

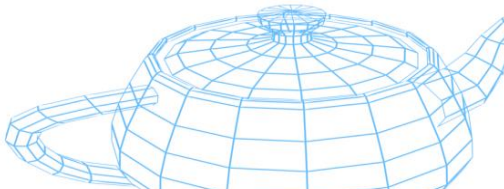


OBJ

- **.obj**: stores information about geometry (vertices, normal vectors, texture coordinates, etc.).

- **.mtl**: optional **M**aterial **T**emplate **L**ibrary file containing data related to the used materials.





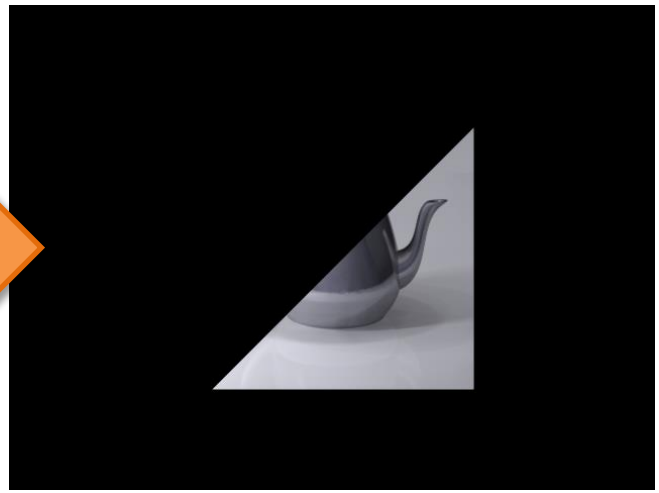
OBJ

```
mtllib example.mtl

# Vertex coords (xyz):
v  -8.0 -8.0 0.0
v   8.0 -8.0 0.0
v   8.0 8.0 0.0

# Tex. coords (uvw):
vt 0.0 0.0 0.0
vt 1.0 0.0 0.0
vt 1.0 1.0 0.0

# Normal vectors (xyz):
vn 0.0 0.0 1.0

# Object:
g SingleTris
usemtl Default
f 1/1/1 2/2/1 3/3/1
```

(file: example.obj)

Material property file

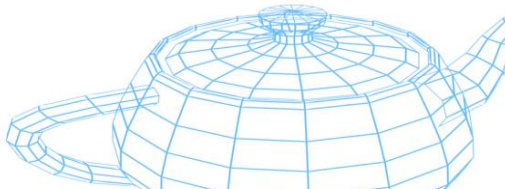
Comment (#)



Object name

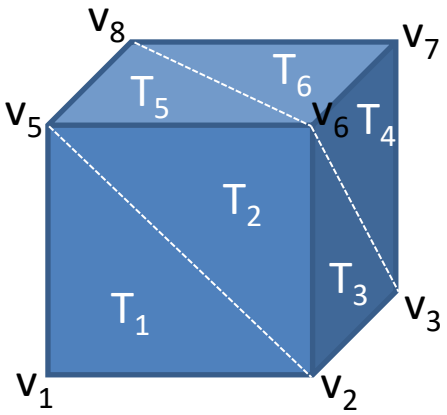
Material name

Face (triangle) data as
vertex/texCoord/normal array IDs

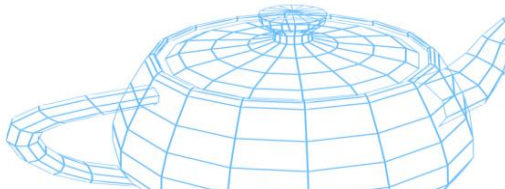


Face index arrays

Vertices = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$
Triangles = $\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}\}$



$T_1 = v_1, v_2, v_5$
$T_2 = v_5, v_2, v_6$
$T_3 = v_6, v_2, v_3$
$T_4 = v_6, v_3, v_7$
$T_5 = v_8, v_5, v_6$
$T_6 = v_8, v_6, v_7$
...



MTL

```
newmtl Default
       Ns 10.0
       Ni 1.5
       d 1.0
       Tr 0.0
       Tf 1.0 1.0 1.0
       illum 2
       Ka 1.0 0.0 0.0
       Kd 1.0 0.0 0.0
       Ks 0.2 0.2 0.2
       Ke 0.0 0.0 0.0
       map_Kd teapot.tga
```

(file: example.mtl)

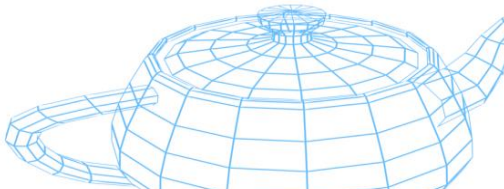
Material name

Specular coefficient

Transparency (0 invisible, 1 opaque)

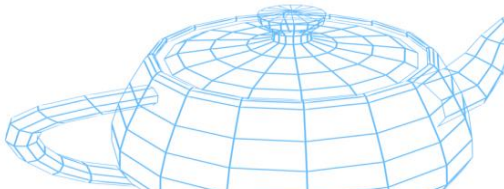
Ambient, diffuse, specular and emissive terms (as seen in the Blinn-Phong equation)

Diffuse texture map file name



VRML

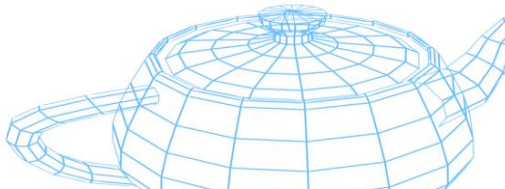
- **V**irtual **R**eality **M**odeling **L**anguage (1995).
- Extension used: **.wrl**
- Half-way between a textual format and a scripting language.
- It evolved into the more recent X3D, an XML-based format.



VRML

```
DEF Plane001 Transform {
 translation 0 0 0
 rotation -1 0 0 -1.57
 children [
  Shape {
   appearance Appearance {
    material Material {
     diffuseColor 1 0 0
     ambientIntensity 1.0
     specularColor 0 0 0
     shininess 0.145
     transparency 0
    }
    texture ImageTexture {
     url "teapot.tga"
    }
   }
```

```
   geometry DEF Plane001-FACES IndexedFaceSet {
        ccw TRUE
        solid TRUE
        coord DEF Plane001-COORD Coordinate { point [
         -8 0 8, 8 0 8, 8 0 -8]
        }
        normal Normal { vector [
         0 1 0, ] }
        normalPerVertex TRUE
        texCoord DEF Plane001-TEXCOORD
   TextureCoordinate { point [
         0 0, 1 0, 1 1]
        }
        coordIndex [
         0, 1, 2, -1]
        texCoordIndex [
         0, 1, 2, -1]
        normalIndex [
         0, 0, 0, -1, ]
        }
     } ] }
```

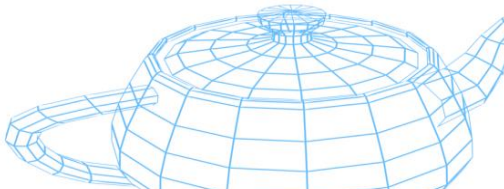


(file: example.wrl)

COLLADA

- **COLL**aborative **D**esign **A**ctivity.
- Open standard (ISO) introduced by Sony and now maintained by the Khronos group (2004).
- Textual XML-based format.
- Usually stored as **.dae** (**D**igital **A**sset **E**xchange).
- https://www.khronos.org/collada/

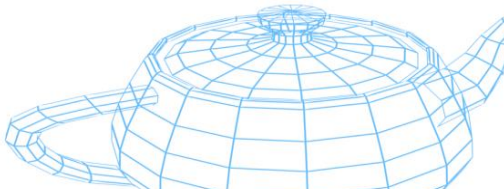
(see *example.dae*)



FBX

- **F**ilm**B**o**X**.
- Invented by Kaydara around 1996, then acquired by Autodesk (2006).
- Industry/commercial standard.
- Both ASCII and binary formats are supported.
- Extension used: **.fbx**
- Autodesk provides a free C++ SDK for working with FBX files.

(see *example.fbx*)



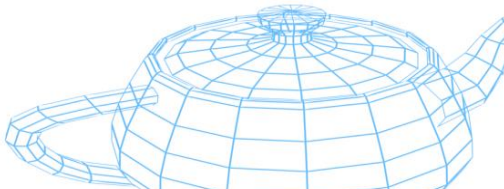
3DS

- Introduced by Autodesk as the native format for **3D S**tudio (DOS version, 1990).

- Binary format made of chunks:
  - A chunk is a binary block of data defined by an ID and a given size (somehow like an XML block):
    - If you do not recognize/need one chunk, you can easily skip it.

- Extension used: **.3ds**

- It includes lights, meshes, materials, animations, and the scene graph:
  - Data is stored with *y* as depth and *z* as height.

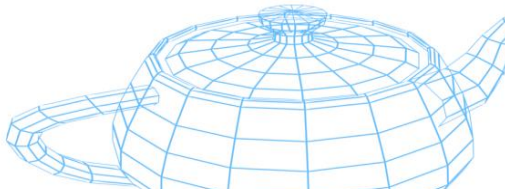


3DS

```
0x4D4D // Main Chunk
├─ 0x0002 // M3D Version
├─ 0x3D3D // 3D Editor Chunk
│   ├─ 0x4000 // Object Block
│   │   ├─ 0x4100 // Triangular Mesh
│   │   │   ├─ 0x4110 // Vertices List
│   │   │   ├─ 0x4120 // Faces Description
│   │   │   │   ├─ 0x4130 // Faces Material
│   │   │   │   └─ 0x4150 // Smoothing Group List
│   │   │   ├─ 0x4140 // Mapping Coordinates List
│   │   │   └─ 0x4160 // Local Coordinates System
│   │   ├─ 0x4600 // Light
│   │   │   └─ 0x4610 // Spotlight
│   │   └─ 0x4700 // Camera
│   └─ 0xAFFF // Material Block
│       ├─ 0xA000 // Material Name
│       ├─ 0xA010 // Ambient Color
│       ├─ 0xA020 // Diffuse Color
│       ├─ 0xA030 // Specular Color
│       ├─ 0xA200 // Texture Map 1
│       ├─ 0xA230 // Bump Map
│       └─ 0xA220 // Reflection Map
│           │  /* Sub Chunks For Each Map */
│           ├─ 0xA300 // Mapping Filename
│           └─ 0xA351 // Mapping Parameters
└─ 0xB000 // Keyframer Chunk
    ├─ 0xB002 // Mesh Information Block
    ├─ 0xB007 // Spot Light Information Block
    └─ 0xB008 // Frames (Start and End)
        ├─ 0xB010 // Object Name
        ├─ 0xB013 // Object Pivot Point
        ├─ 0xB020 // Position Track
        ├─ 0xB021 // Rotation Track
        ├─ 0xB022 // Scale Track
        └─ 0xB030 // Hierarchy Position
```

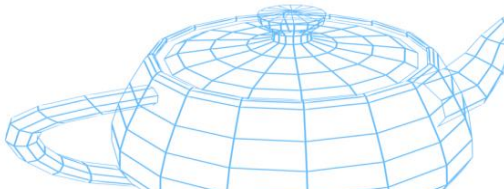
Game file formats

- Several videogames allow users to add/edit/replace/personalize the game content (a.k.a., “modding”):
  - File formats are often documented or reverse-engineered.
  - Additional tools are provided, such as importers/exporters for common 3D editors.

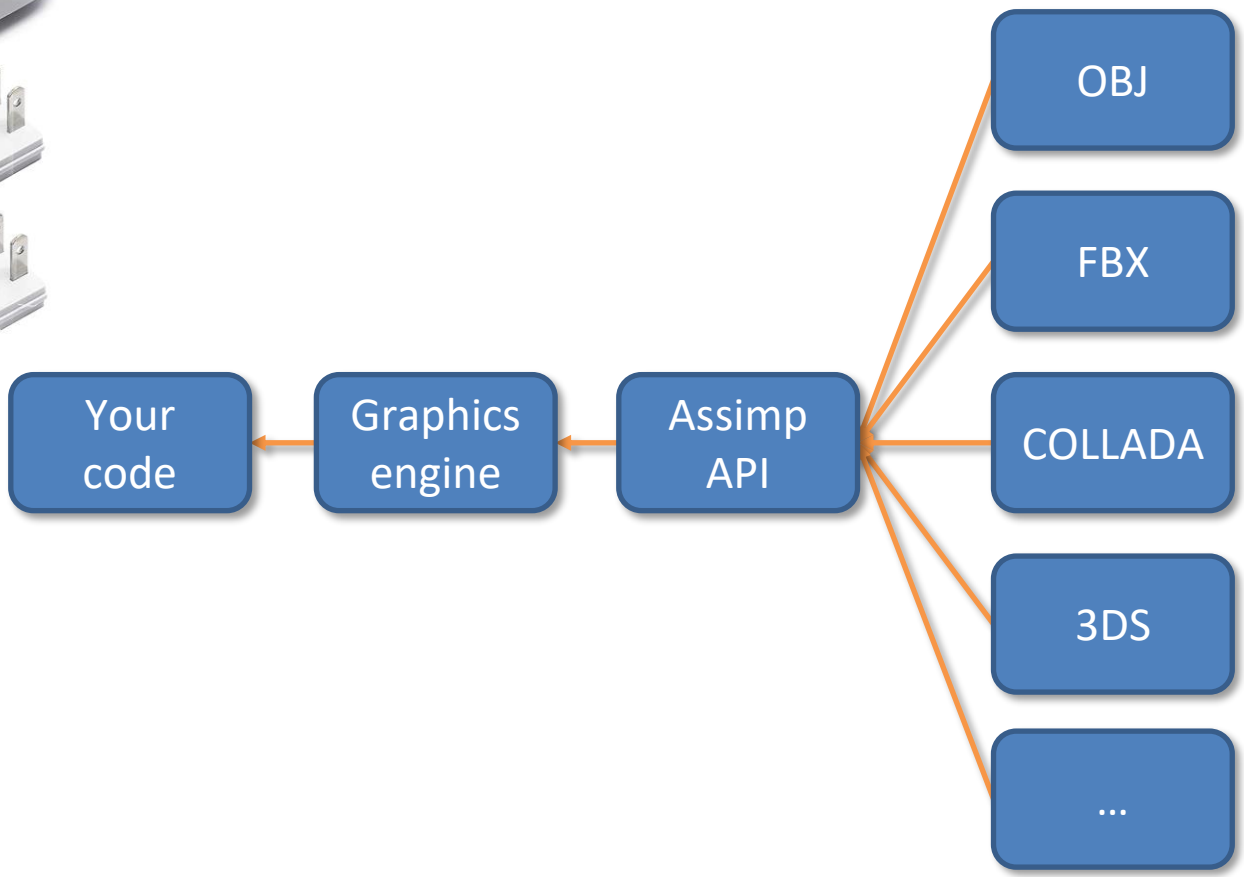


Game file formats

- MD2/3/5, WAD:
  - Introduced by ID Software in their games (Quake, Doom, etc.).
- WAD was used in Doom to store one entire level map (including geometry, textures and additional media).
- MD5MESH contains the mesh data, while MD5ANIM contains animations:
  - Supports skeletal animation.
  - Textual format.
  - http://tfc.duke.free.fr/coding/md5-specs-en.html

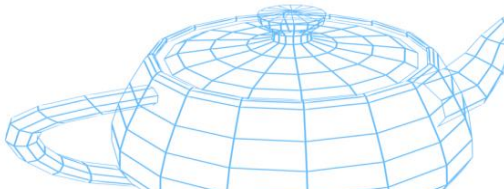


ASSIMP - Asset Import Library



Assimp

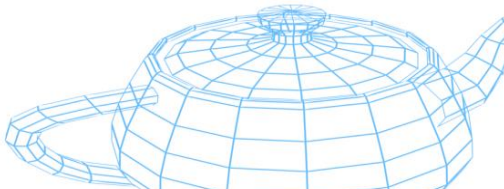
- C/C++ API supporting a wide range of 3D file formats.
- Works both under Windows and Linux.
- Open-source, released under the BSD license.
- Available at: http://www.assimp.org/



Custom formats

- When an ideal format is not available, many software developers adopt their own file format:
  - Performance: data is preprocessed to perfectly match the engine structures.
  - Compactness: only information really used by the engine is stored.

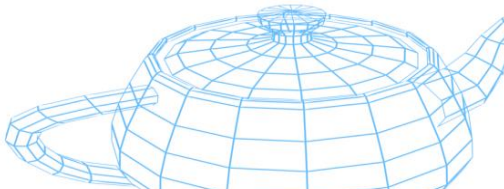
- Defining a custom file format is relatively trivial, the problem is importing/converting data:
  - …as with any other file format…



Custom formats

- Custom formats can be the output of a custom converter:
  - E.g., by using a library to create a command-line application parsing a (complex) file format to output a (simpler) binary file.

- Custom formats can be the output of an existing commercial software:
  - E.g., writing a plugin for embedding your own format into an existing 3D editor:
    - Clean way when you must work with 3D artists and designers or when you create a professional graphics engine.



OVO

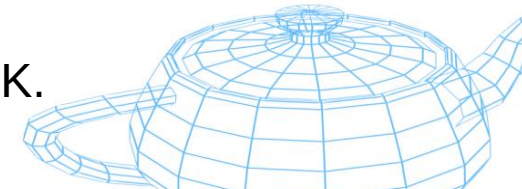
- **O**ver**V**ision **O**bject.

- Custom file format used by the OverVision graphics engine.

- Several advantages:
  - Directly integrated in 3D Studio Max through a plugin.
  - OpenGL-friendly (same conventions and data-types).
  - Includes nodes, materials, textures, meshes, lights, and more.
  - Includes additional information, such as mesh bounding boxes, radii, targets.
  - Automatically converts textures into power-of-two .dds files.

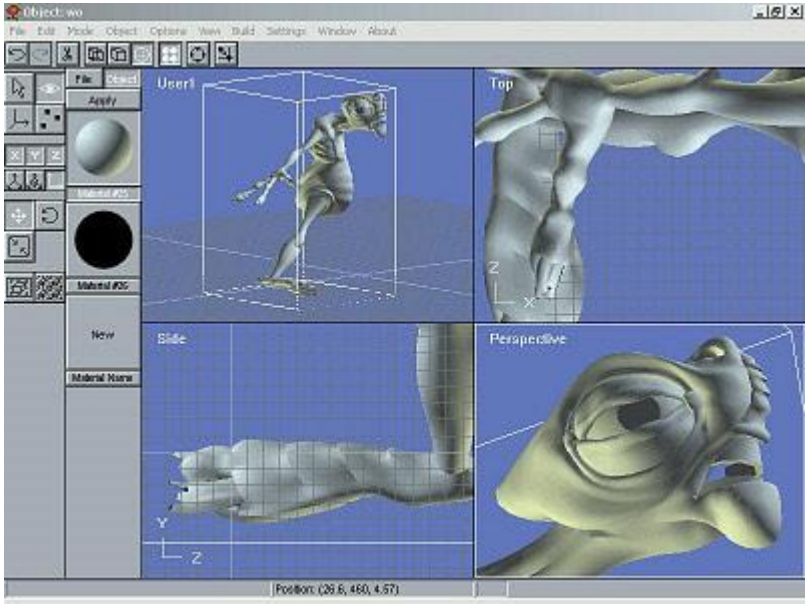
- Binary, chunk-based format.

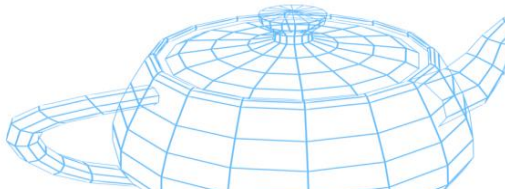
- See available documentation and examples in the OVO SDK.



3D file editors

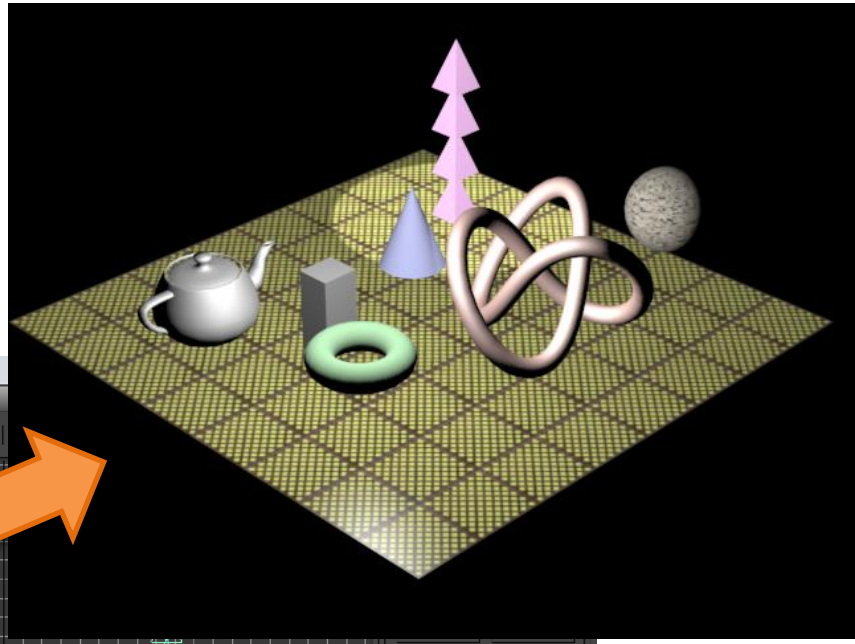
- Commercial software:
  - 3D Studio Max
  - Maya
  - Cinema 4D
  - LightWave
  - …
- Free editors:
  - Blender
  - Anim8or
  - Milkshape 3D
  - …

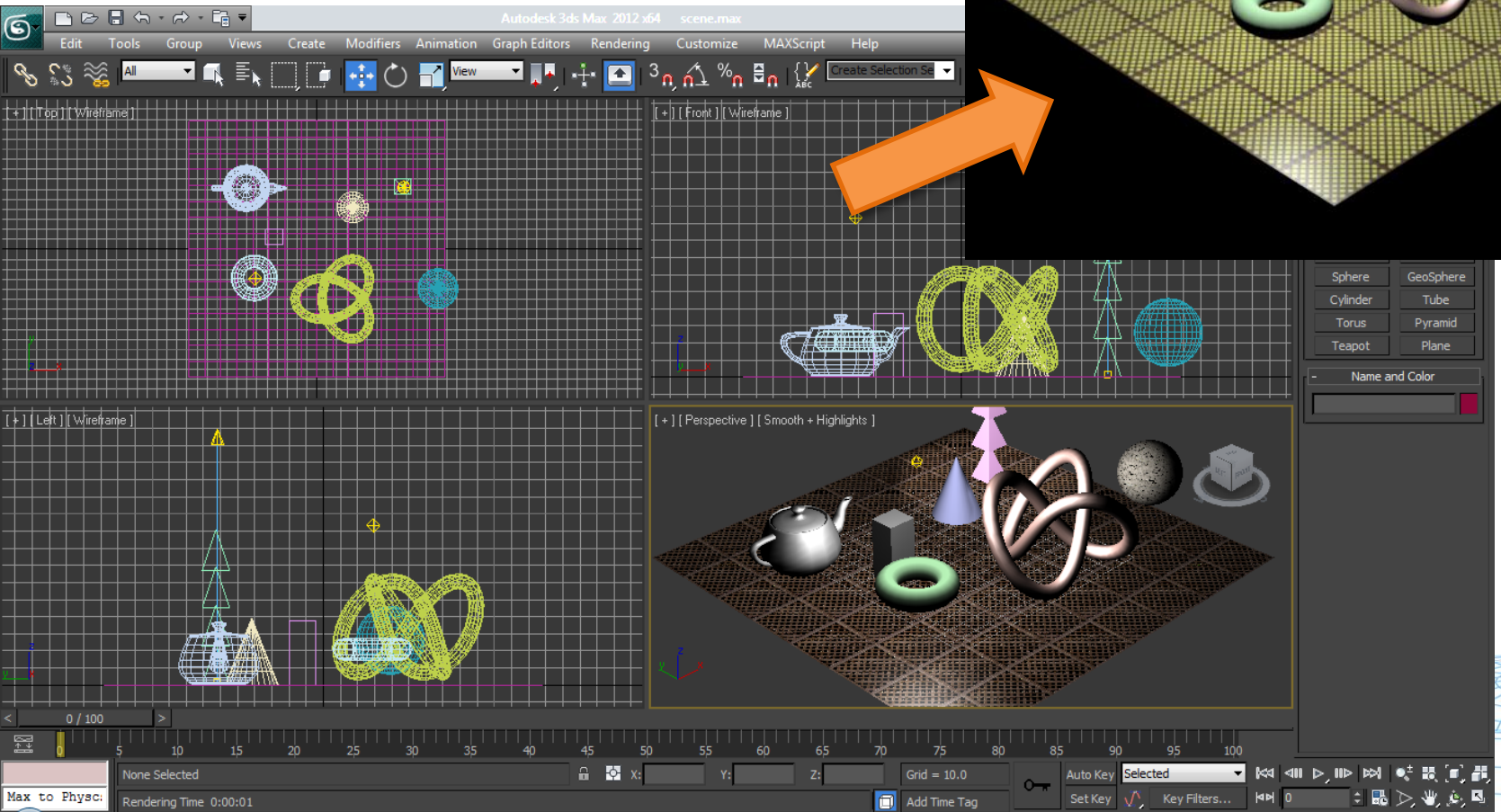




3D scene example

(See scene. files)*






Tutorial
OVO SDK + 3dsmax
VIDEO
DISPLAY