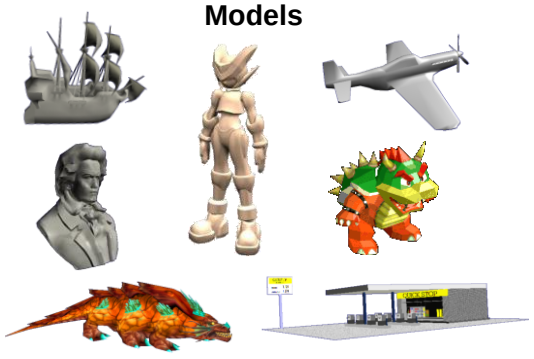
CSc 165
Computer Game Architecture

11 – 3D Modeling for Games

Overview

- **Model Characteristics**
- **3D Model File Formats**
- **Model Loaders**
- **Digital Content Creation (DCC) Tools**
- **Skinning and UV-unwrapping**

Models



Static Data

- 3D geometry (vertex data)
- Polygon (face) data
- Rendering attributes
  - Wireframe / Faceted / Smooth-shaded
  - Lighting & Materials
- Texturing (“skinning”) data

Animation Data (sometimes)

- Model structure (skeletons, joints)
- Model poses
- Animation sequences
  - walk / run / jump / die …

Common 3D Model File Formats

.3ds – 3D Studio Max format
.blend – Blender format
.dae – COLLADA Digital Asset Exchange format
.dem – USGS Standard for Digital Elevation Models
.dxf – Autodesk's AutoCAD format
.hdf – Hierarchical Data Format
.iges – Initial Graphics Exchange Specification
.iv – Open Inventor File Format Info
.lwlo, .lwob & .lwsc – Lightwave 3D file formats
.md2/.md3/.md4/.md5 – Quake Model Files
.ms3d – Milkshape 3D binary format

.msdl – Manchester Scene Description Language
.nff & .enff – (Extended) Neutral File Format
.obj – Alias|Wavefront Object Files
.off – 3D mesh Object File Format
.oogl – Object Oriented Graphics Library
.ply – Stanford Scanning Repository format
.pov – Persistence of Vision ray-tracer
.qd3d – Apple's QuickDraw 3D metafile format
.rkm – RAGE sKeletal Mesh
.viz – used by Division's dVS/dVISE
.vrml – Virtual Reality Modeling Language
.x – Microsoft's DirectX/Direct3D file format
.x3d – eXtensible 3D XML-based scene description format

See wikipedia – list of file formats

.OBJ File Commands

Vertex data
- **v** – geometric data
- **vt** – texture data
- **vn** – vertex normals

Elements
- **p** – point
- **l** – line
- **f** – face
- **curv** – curve
- **surf** – surface

Grouping
- **g** – group name
- **s** – smoothing group
- **mg** – merging group
- **o** – object name

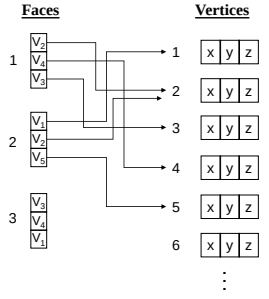
Render Attributes
- **usemtl** – material name
- **mtllib** – material file name
- **lod** – level of detail
- **shadow_obj** – shadow casting

.OBJ Example

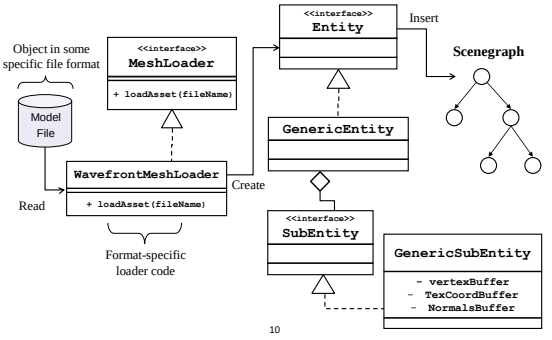
```
# File: 'cube.obj'
# This file uses "OBJ" format to define a cube
# Define 8 cube vertices
v  -1.0  -1.0  -1.0
v  -1.0  -1.0   1.0
v  -1.0   1.0  -1.0
v  -1.0   1.0   1.0
v   1.0  -1.0  -1.0
v   1.0  -1.0   1.0
v   1.0   1.0  -1.0
v   1.0   1.0   1.0

# Specify the file (library) containing materials
mtllib cube.mtl

#continued...
```


OBJ Face/Vertex Data Structures

```
# File 'man.obj'
v  -1.0  -1.0  -1.0
v  -1.0  -1.0   1.0
v  -1.0   1.0  -1.0
...
vt 0.72 0.32
vt 0.86 0.33
...
vn 1.0 0.0 1.0
vn -1.0 0.0 0.5
...
f 2/1/1 4/2/1 3/3/2
f 1/4/2 2/4/2 5/5/3
...
```



Content Loaders in RAGE

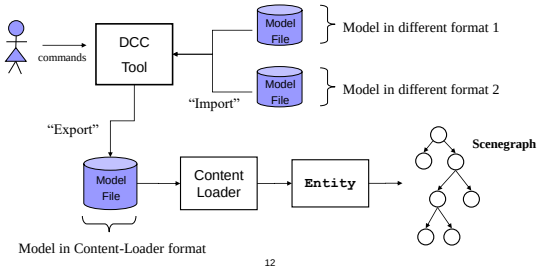


Example: .OBJ Content Loader

```
public class WavefrontMeshLoader implements MeshLoader
{   ...

    public void loadAsset(Mesh mesh, Path path)
    {   List<Float> verts = new ArrayList<>();
        List<Float> norms = new ArrayList<>();
        List<Float> texcoords = new ArrayList<>();
        BufferedReader br = Files.newBufferedReader(path);
        String line;
        while ((line = br.readLine()) != null)
        {  if (line.startsWith(VERTEX_COMMAND))
           {  processVertexPositionCommand(line, verts); }
           else if (line.startsWith(TEXTURE_COORD_COMMAND))
           {  processVertexTextureCommand(line, texcoords); }
           else if (line.startsWith(NORMAL_COMMAND))
           {  processVertexNormalCommand(line, norms); }
           else if (line.startsWith(FACE_COMMAND))
           {  processVertexFaceCommand(line); }
           else if (line.startsWith(MATERIAL_COMMAND))
           {  materialLib = line.split(" ")[1]; }
        }
        SubMesh sm = mesh.createSubMesh(mesh.getName());
        createVertexBuffers(verts, norms, texcoords);
        sm.setVertexBuffer(toFloatBuffer(vertexPositionsList));
        sm.setNormalBuffer(toFloatBuffer(vertexNormalsList));
        sm.setTextureCoordBuffer(toFloatBuffer(vertexTexCoordsList));
        sm.setIndexBuffer(toIntBuffer(vertexIndicesList));
    }
...
```


Digital Content Creation (DCC) Tools

Too much work to do "by hand"



DCC Tools

Commercial

- Maya
- Houdini
- Modo
- Lightwave
- etc., etc. …

Free (or nearly)

- Blender
- SketchUp
- Maya / 3DStudio Max for students
- etc., etc., …

Blender

Features:

- Modeling, texturing, lighting, UV-mapping
- Animation rigging/skinning
- Particle, physics, soft-body, fluid & cloth simulations
- Game engine, video post-processing
- Python scripting/plugin support
- Cross-platform (Windows, Linux, Mac OSX, FreeBSD, Solaris….)
- Import/export for a wide variety of model file formats
- Extensive online documentation/wiki support

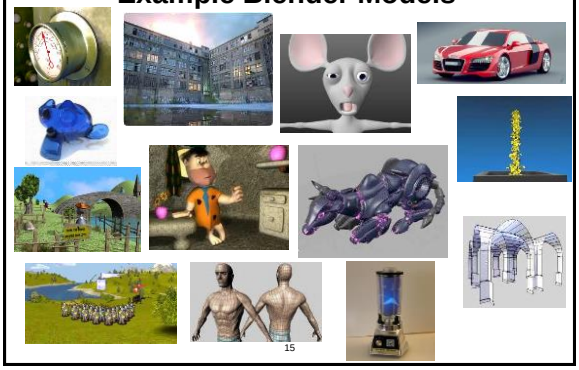
Availability:

- Free (GNU GPL license), at **`http://www.blender.org`**

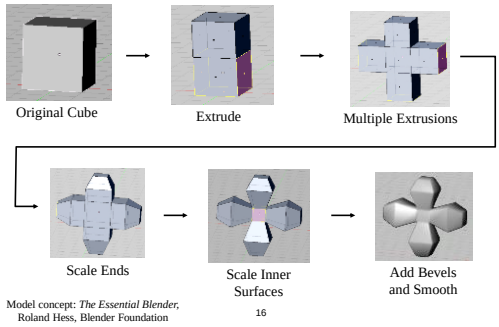
Tutorials:

- **`http://www.blender.org/support/tutorials`**

Example Blender Models



Building Models From Primitives



Model concept: *The Essential Blender,* Roland Hess, Blender Foundation

Model Exporting

Most DCC tools export a variety of formats

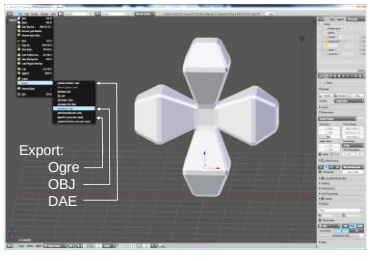
Blender supports exporting to:

- Collada XML (".dae")
- Wavefront OBJ (".obj")
- Stanford Graphics Library (".ply")
- 3DStudio Max (".3ds")
- A host of others – and more via "add-ons"

RAGE has ModelLoaders for

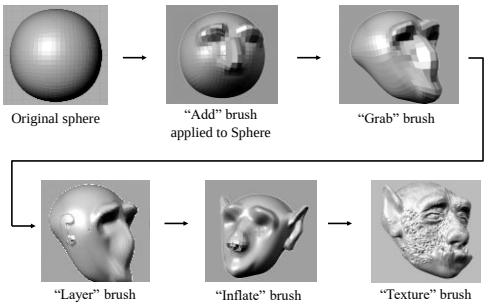
- OBJ
- RAGE Skeletal Animation *(Blender exporter add-on)*

Exporting Models




```
# Blender3D v249 OBJ File:
CrossDemo.blend
# www.blender3d.org
mtllib CrossDemo.mtl
v 0.515410 2.745096 -0.496360
v 0.515410 2.745096 0.496359
v -0.477309 2.745096 0.496359
v -0.477309 2.745096 -0.496360
v 0.515410 3.737815 -0.496359
v 0.515410 3.737815 0.496360
v -0.477309 3.737815 0.496359
v -0.477309 3.737815 -0.496360
v -0.523100 6.135959 0.542150
v -0.523100 6.135958 -0.542150
v 0.561200 6.135959 0.542150
v 0.561201 6.135958 -0.542150
...#total 44 vertices
usemtl Material
s 1
f 2 6 7
f 5 1 4
f 12 10 9
f 13 14 15
f 1 42 44
f 21 24 44
...#total 42 faces
```

Advanced Modeling: Sculpting



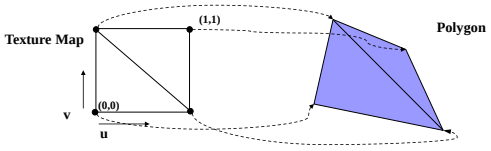
Images from: *The Essential Blender*, Roland Hess, Blender Foundation

Skinning

Applying "texture" to 3D models

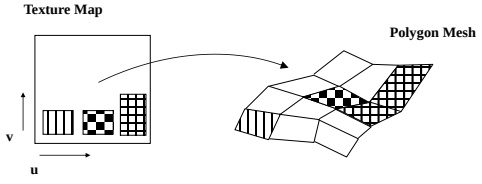
Problem: texture mapping is per-polygon:

Models are *collections* of polygons



Texture Space Subdivision

Texturing *meshes* by *dividing* texture space:



Skinning Example

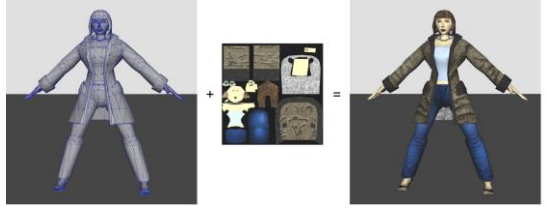


Image credit: Practical Java Game Programming, Clingman et.al., Charles River Media

Texture Subdivision Difficulties

Requires creating a complex mapping
(and the model is often curved)

- How do we create the texture map?
- How do we determine the correct mapping?

Mapping texture locations to model coordinates

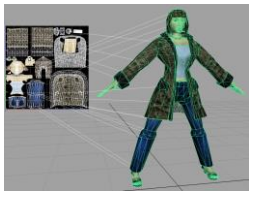


Image credit: Practical Java Game Programming, Clingman et.al., Charles River Media

Creating Texture Skins

- Create the model
- *Flatten* the model ("***UV Unwrapping***")
  - Mark seams
  - Cut along seams ("project")
- Save unwrapped (flattened) UV image
- Save model with texture coordinates
- Use UV image as a "pattern" to create tex map
  - GIMP, PhotoShop, Paint, ...
- Load resulting texture image into game

Example: Soup Can

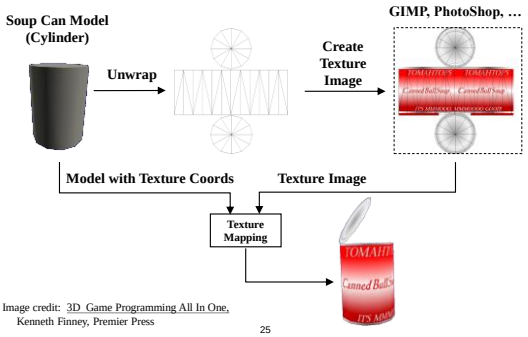
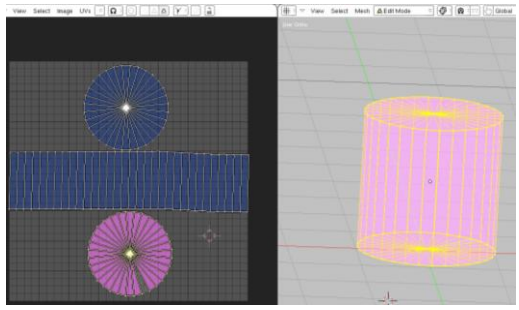


Image credit: 3D Game Programming All In One, Kenneth Finney, Premier Press

Blender UV Unwrapping



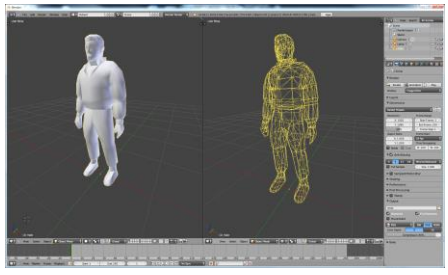
Character Models

More complex, but same approach:

- Create model
- Mark/cut seams (some tools support *groups)*
- Project UV's
- Save projection image
- Use projection image to create texture
- Apply texture to UV map/model

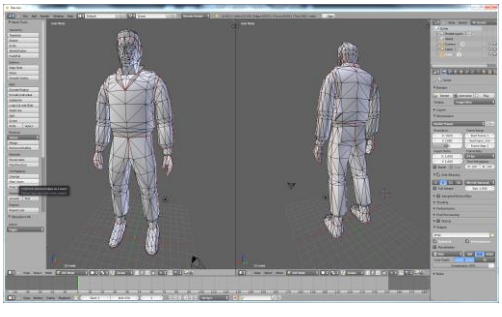
Blender: Character Example

Step 1: Load/Create Model

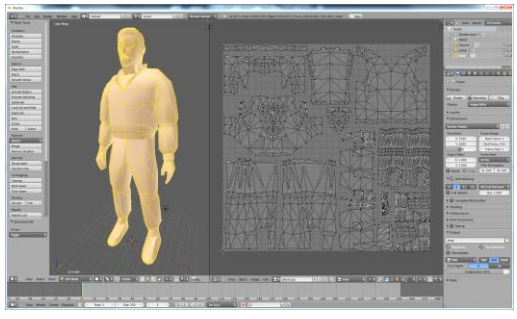


Example from: 3D Game Programming All In One, Kenneth Finney, Premier Press

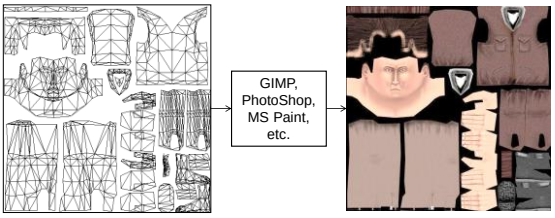
Step 2: Mark Seams



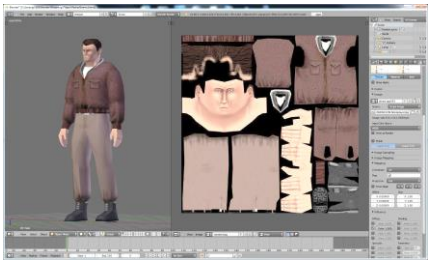
Step 3: Export UVs



Step 4: Paint Texture



Step 5: Apply Texture to Model



…either in Blender, or directly in RAGE