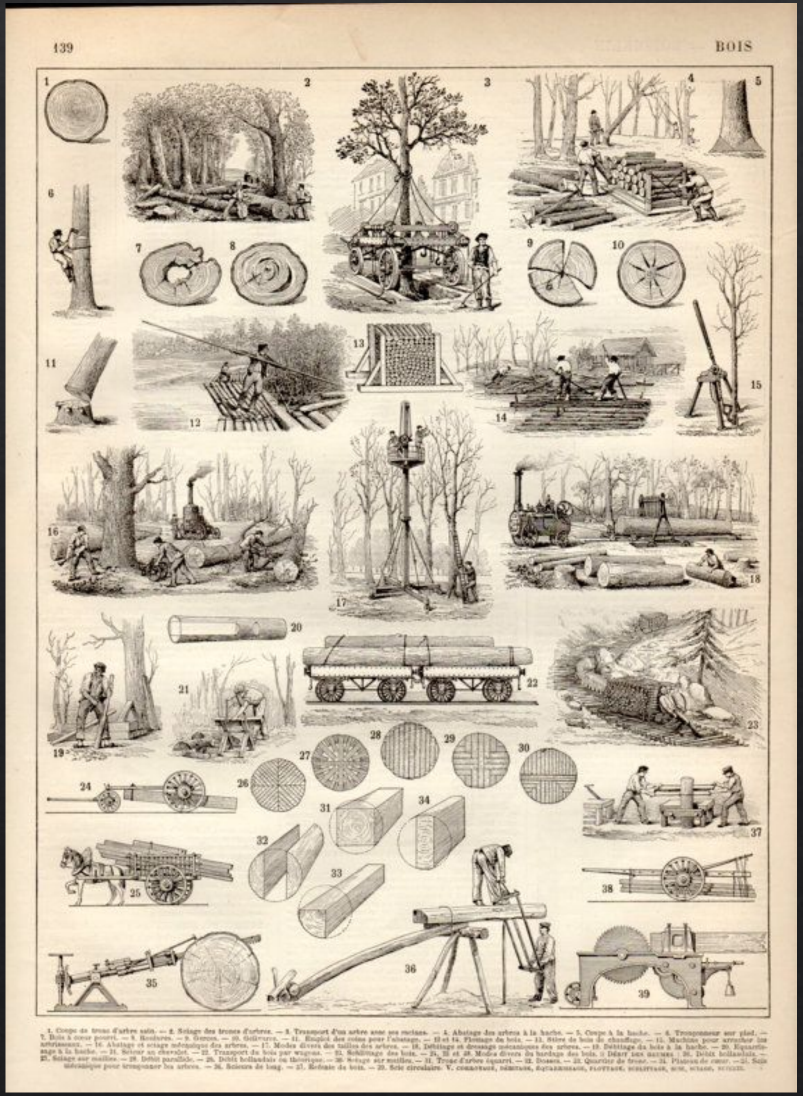
ETGG3802

Lecture2: Logging



How’s it used in U/U

- Unity:
  - print statement
- Unreal
  - (C++) write to console output
  - *On-Screen text
- Goals of both:
  - Display DEBUG information to developer
  - Doesn’t replace the need for debuggers!
- Additional goal for us: write to a file
  - Supplements ogre’s log, but much smaller

Ogre Overlays

- 2D hierarchical elements, but super low-level
  - Relative (0-1) or Pixel
- Basically
  - An Overlay – contains 1 or more panels
  - Containers (Panels)
    - Materials
  - TextArea – dynamically rendered text
- No “Buttons”, “Drop-down boxes”, etc.
  - But DIY is definitely an option!
- Two ways to make them:
  - With a .overlay script file (we might see these later)
  - In C++ code (this lab)

Ogre Overlays, cont.

- Good starting reference: http://wiki.ogre3d.org/Creating+Overlays+via+Code
- Include files
  - all are in the $OgreSDK\include\OGRE\Overlay sub-folder
  - We should already have $OgreSDK\include\OGRE as a standard location
  - So…all includes will look like #include<Overlay/the_file.h>
  - The files:
    - OgreOverlay.h
    - OgreTextAreaOverlayElement.h
    - OgreOverlayManager.h
    - OgreOverlayContainer.h
    - OgreOverlaySystem.h

Ogre Overlays, cont.

- Minor Notes
  - add this line to your Application's startup routine

```
mSceneManager->addRenderQueueListener(getOverlaySystem());
```

    - getOverlaySystem gets the overlay system that ApplicationContext set up.
    - addRenderQueueListener makes the main render routine check for overlays our application creates.
  - When setting font for your TextArea
    - Define your own (see $OgreSDK\bin\Media\packs\SDKTrays.zip, the .fontdef in particular)
    - Use **SdkTrays/Value** (already in the above pack)
  - There's a setColour method of TextArea elements (as well as setColourTop and setColourBottom).