

HCI 558 X

Lecture 14:

April. 10, 2007

- Quiz 3 return
- Python scripting: regular grids + triangles



897390570
080280850
567847298
217898438
897594797

- I: The 3s' color (hue) can be “caught” by a pre-attentive mechanism (step I: feature extraction), 7s need attention- so we need to go through ALL numbers, assemble a pattern and compare it to a 7, this imposes a cognitive load.

Feature extraction is fast because it uses automatic, very parallel processing (“hardware”) and has much higher bandwidth than the attentive process. Other feature extraction mechanisms are shape, orientation, movement, size, convex/concave, blur, ...

- 2: Computer graphics uses similar principles to create shapes of objects that our pattern/shape recognition mechanisms use to recognize shapes and properties of real-world objects. The main principles are: color “shades” (from light), specular highlights, textures, perspective, shadows.
- 3: Change blindness studies suggest that the high level human visual system is very different from a camera-film model. We lose major parts of what should be stored on film (memory) by just looking at an empty screen for a split second, hence our vision cannot be based on persistent storage. It rather is like a dynamic, always ongoing construction project where “unimportant” objects are lost, unless pointers to them have been created by attention.

Python

- Recap: echo6.py - dealing with header information:
- look at 2Dgrid_w_header.txt
- skip certain lines (1. and 3.) with continue
- in 2. line: get width and height of 2D grid
- write into output file out.txt
- ' '.join(list) : create space sep. string from list
- ['1', '2', '3'] is joined to "1 2 3"



Irregular data



- copy hci558/data/irregular to your folder
- points (vertices) - (lines) - triangles (quads) - mesh
- Conversion to dx?
- Look at cube.stl and tris.wrl
- cube.stl: 3 vertices (position) define triangles (data?)
- tris.wrl:
- keywords: points (x y z) and coordIndex (a b c -1)

recap - points, index, triangles

- points = positions
- index = connections (= triangles!)
- -1 : end of polygon
- data ? dep? ref ?
- python script: get <stuff>
- point [<stuff>]
- coordIndex [<stuff> -1]
- keep track of numbers of points and triangles for later
- need to write <stuff> and dx objects descriptions

```
coord Coordinate{
  point [
    0.000000 0.100000 -2.200000
    1.000000 0.200000 0.300000
    2.000000 0.100000 -0.200000
    ...
    3.000000 0.100000 2.200000
    point #7
    0.000000 1.300000 0.700000
    1.000000 1.100000 -0.100000
    2.000000 2.200000 0.000000
    3.000000 2.100000 -1.700000
  ]
}
coordIndex [
  0 5 4 -1
  2 7 6 -1
  1 2 5 -1
  2 5 6 -1
  0 1 5 -1
  2 3 7 -1
  5 6 4 -1
]
```

point # 0

point #7

triangle # 0

triangle #5

dx triangle format

- look at tris.dx - object descriptions (= components)
- object 1: 8 points (vertices) = positions (x y z)
- object 2: 5 triangles (a b c as index to positions)
- index starts at 0, element type is triangles (!), ref ?
- object 3: data, one for each position (8) dep positions
- object 4: data, per connection = per triangle (dep connections)
- object “trimesh”: makes field from components
- data follows: can also be in separate file
- geometry from stl or wrl file (+ data)

- in your irregular folder: dx -edit tris1.net
- connection or point dependent?
- how to change ? look into .dx file
- tris2.net
- tris3.net (mark ? numbers?)
- tets.net (tetrahedrons = “3D triangles”)
- hardware/software rendering, change planes

- Questions?
- made copies of Quiz 2 from everybody?
- rest of midterm reports / comments this week
- HW 6 - results (+ other HW) on course web page ?
- next lecture: recap session - everybody sent me at least two questions
- Thursday: up to you - talk about: HW 6, python, import issues, dx file format,

- 2D_gridA.py
- write a dx 2D grid header first
- use width and height in gridpositions and gridconnection object
- warning about order of dimensions:
- .dx file: <slow varying>, <fast varying> !!!
- here: height, width (Y dimension, X dimension)
- 2D_gridB.py: print cell values as 2D array indices
- try 2Dgrid.net
- bigger example: pic.py (pic.dx)