Lecture 25: Texture Mapping

December 3, 2020

Now to Texture mapping

Image from http://www.minecraftercamp.com

Adding surface detail

- Surfaces in the world have appearance
  - They are seldom one flat color.
  - They have true texture – repeating patterns.
  - They have structured markings.
  - They have tiny changes in surface height.
- Purists view (don't try this!)
  - Use ever more even smaller uniform triangles.
- Pragmatists view
  - Paint surfaces with images – texture mapping.

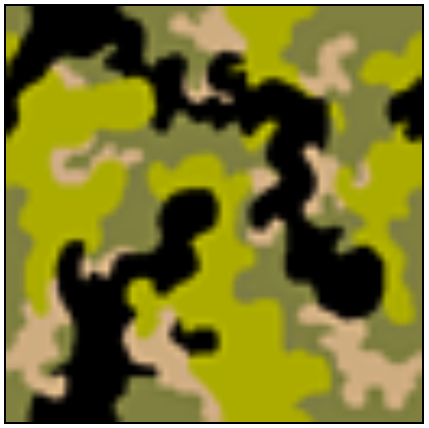
Texture Mapping

- Use projective geometry to compute where vertices appear in the image
- Apply shading to determine the color of pixels
  - -- or –
- Map an existing texture onto a surface
  - Textures supercede/augment the specification of surface material
  - Leaves room for distinction diffuse vs. specular

Mapping

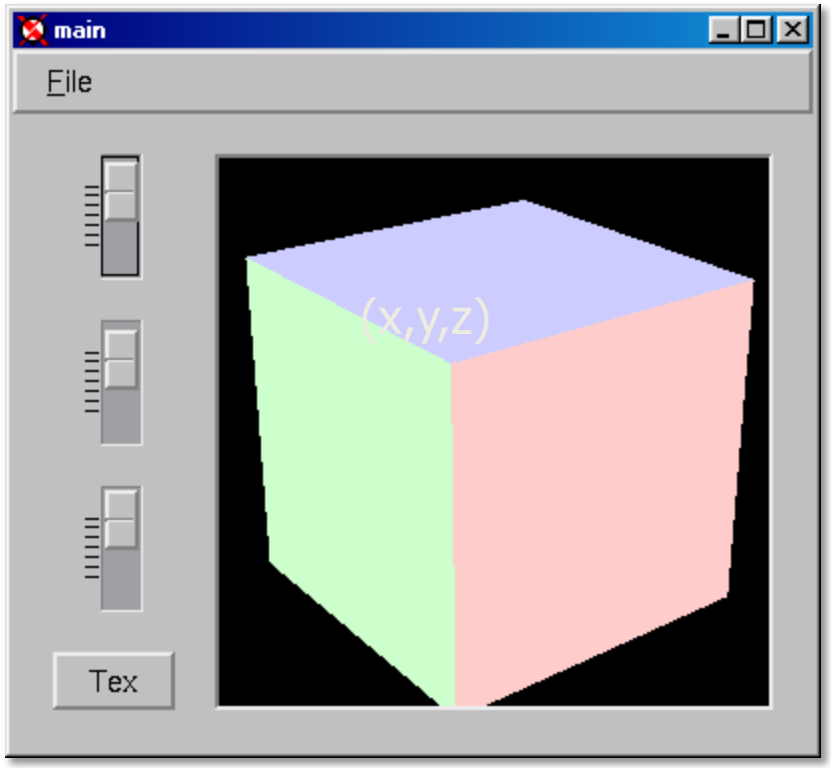
- Guess what? The underlying problem is to apply a geometric transformation

(0,0)



(1,1)

t,s coordinates



Mapping (II)

- Textures are color images
  - Logical texture coords run from (0,0) to (1,1)
  - Coordinates fixed regardless of image size
- Polygons are 2D surfaces in a 3D space
- The transformation from texture coordinates to surface coordinates is expressed as – you guessed it – a matrix

Texture Matrices

- Given vertices and the corresponding texture coordinates…

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- How many correspondences are needed?
- Maps from surface to texture

Worked Example - Before

- Consider point correspondences
  - Pairs of points in texture and 3-D coordinates
  - Three such pairs of points yield six constraints
  - Constraints match free variables – six.
- Specifically
  - Point (0,0) matches point (1,2,3)
  - Point (1,1) matches point (2,2,2)
  - Point (0,1) matches point (3,2,2)

Worked Example

Match #1 $\begin{vmatrix} 0 \\ 0 \end{vmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \end{vmatrix} \begin{vmatrix} 1 \\ 2 \\ 3 \end{vmatrix}$

Match #2 $\begin{vmatrix} 1 \\ 1 \end{vmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \end{vmatrix} \begin{vmatrix} 2 \\ 2 \\ 2 \end{vmatrix}$

$\begin{vmatrix} 0 \\ 1 \end{vmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \end{vmatrix} \begin{vmatrix} 3 \\ 2 \\ 3 \end{vmatrix}$

Match #3

$$0 = a + b + c$$
$$1 = 2a + 2b + 2c$$
$$0 = 3a + 2b + 3c$$

3 equations for first texture coords

Linear Alg. Setup

$$\begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} = \begin{vmatrix} 1 & 2 & 3 \\ 2 & 2 & 2 \\ 3 & 2 & 3 \end{vmatrix} \begin{vmatrix} a \\ b \\ c \end{vmatrix}$$

$$\begin{vmatrix} 0 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 2 & 3 \\ 2 & 2 & 2 \\ 3 & 2 & 3 \end{vmatrix} \begin{vmatrix} d \\ e \\ f \end{vmatrix}$$

Solved in Maple

$a = 0 \quad b = 3/2 \quad c = -1$

$d = \frac{1}{2} \quad e = \frac{1}{2} \quad f = -\frac{1}{2}$

And Easier in SageMath

cocalc.com/projects/522a43e0-1873-4569-b71f-81078b04cbaf/files/Handouts/Ha

CS410 Fall 2020 - CoCalc

Projects | CS410 Fall 2020 | CoCalc | Help | Account

CSU CS410 Fall 2020.course | cs410_2020_lec04n01.ipynb | cs410_Fall2020_Lec25n01.ipyn

File | Notebook

File Edit View Insert Cell Kernel Help | CPU: 0% | Memory: 253MB | Trusted | SageMath 9.2

tab | Code | Validate

1

Texture Coordinates Solve Transform

This is a quick workup of solving for texture coordinates given three pairings. In the next equation the six degree of freedom transformation mapping from a 3D coordinate on a survace and a 2D coordinate in a texture map is expressed using six free variables indicated 'a' through 'f'.

Ross Beveridge, December 3, 2020

In [1]:

```
var('a', 'b', 'c', 'd', 'e', 'f')
var('x', 'y', 'z')
var('u','v')
TM = Matrix(SR, 2,3, ((a,b,c),(d,e,f)))
UV = Matrix(SR, 2,1, ((u),(v)))
PT = Matrix(SR, 3,1, ((x),(y),(z)))
pretty_print(UV, LatexExpr(" = "), TM, PT)
```

0.019 seconds | 2

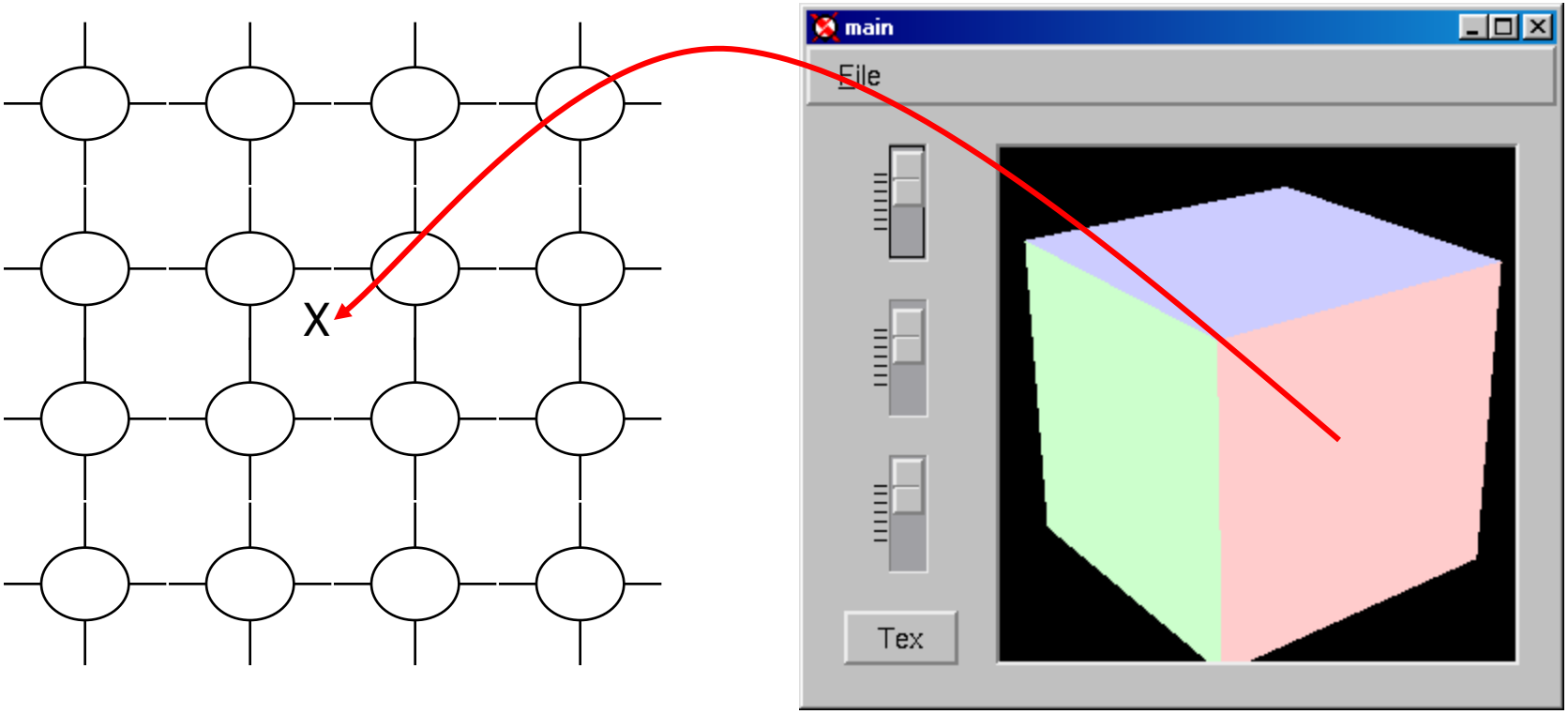
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Texture Mapping (II)

- The fragment processor computes a reflectance color for every pixel
- When textures are enabled,
  - The fragment processor also computes a texture value for every pixel
  - Using the pixel to do texture mapping
- These values are multiplied together to produce the final value

Issue #1: Sampling

- The mapping from surface points to texture coordinates produces real values



Sampling

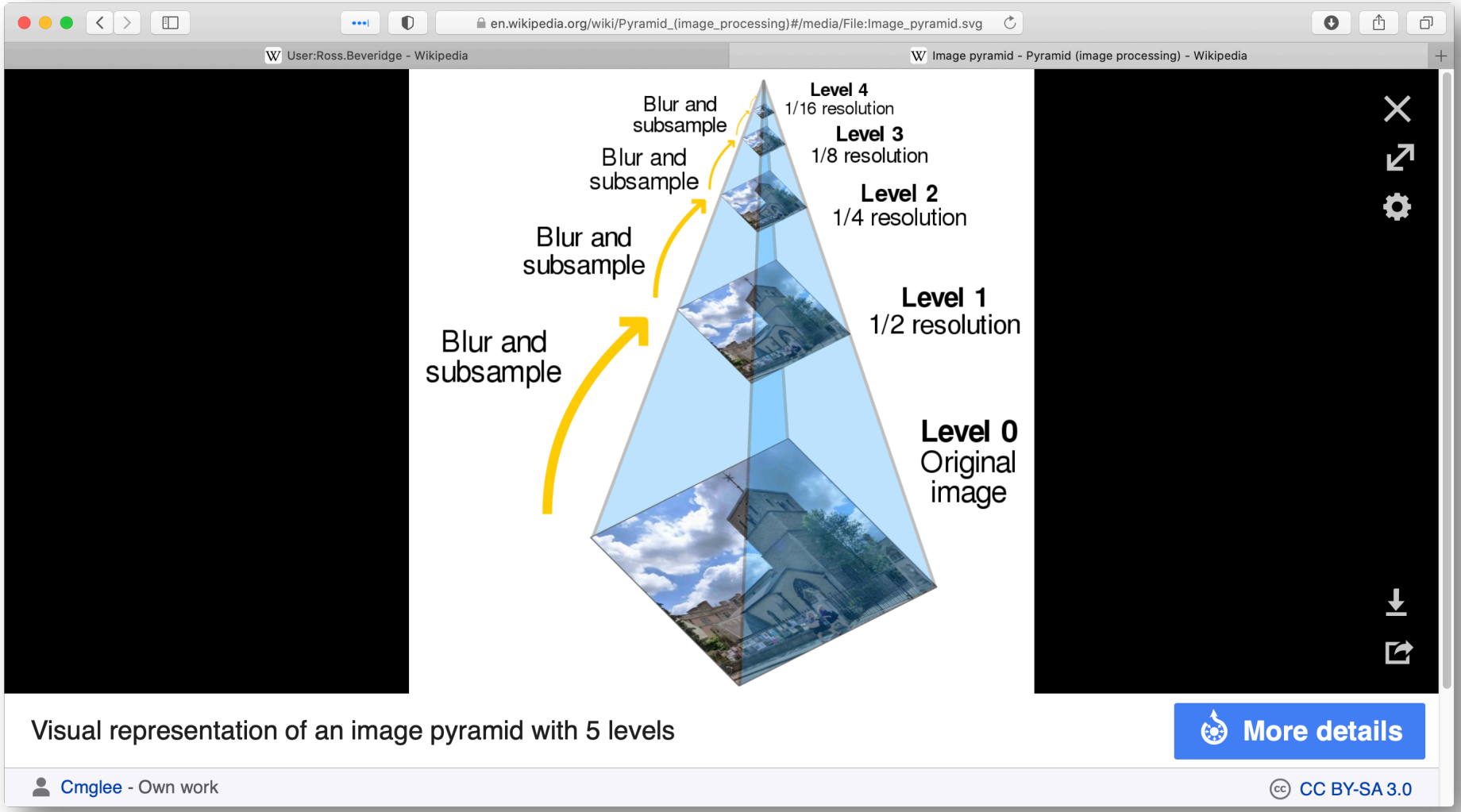
- Nearest-neighbor:
  - pick the closest texture pixel
- Bilinear:
  - linearly interpolate in both dimensions
- Bicubic:
  - fit a 3rd order surface to 16 surrounding points
  - Not as expensive as it sounds

Sampling (III)

- A better solution is for the texture map to be roughly the same size as the surface projection.

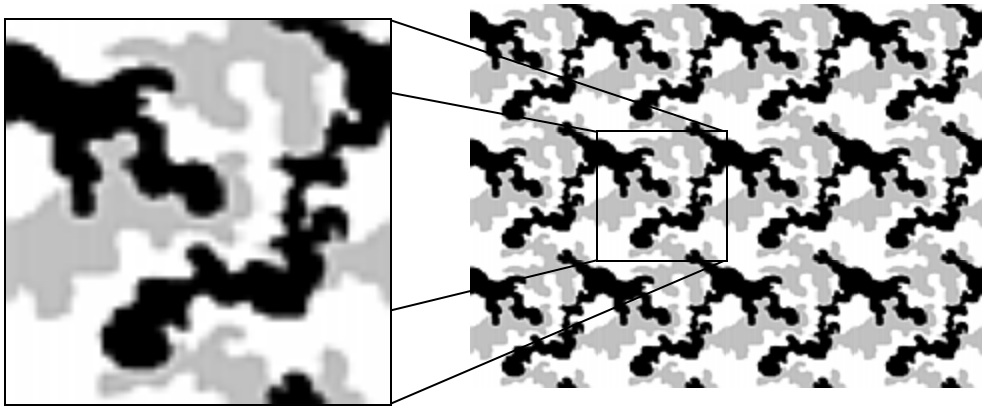
- A MipMap is an image pyramid built from a texture map
  - Example: if the texture is 64x64, the pyramid also includes 32x32, 16x16, etc.

Image Pyramid

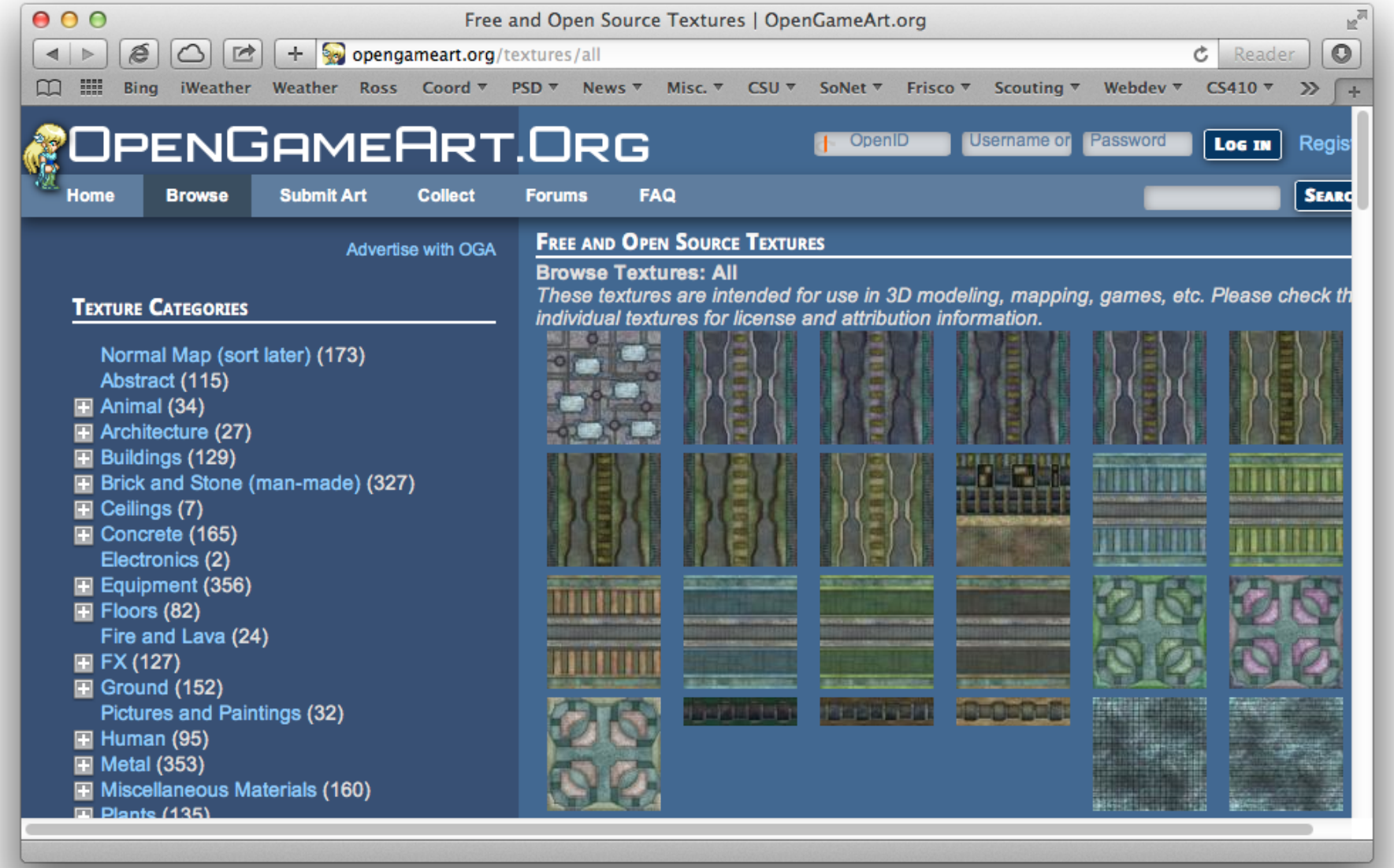


Issue #2: Getting Textures

- WEB!
  - **millions** of textures – people use them for backgrounds of web pages a lot! You can download them in bulk packages, etc..
- Build your own
  - Make them "seamless"
    - When tiled, you cannot see the edges of the tiles.



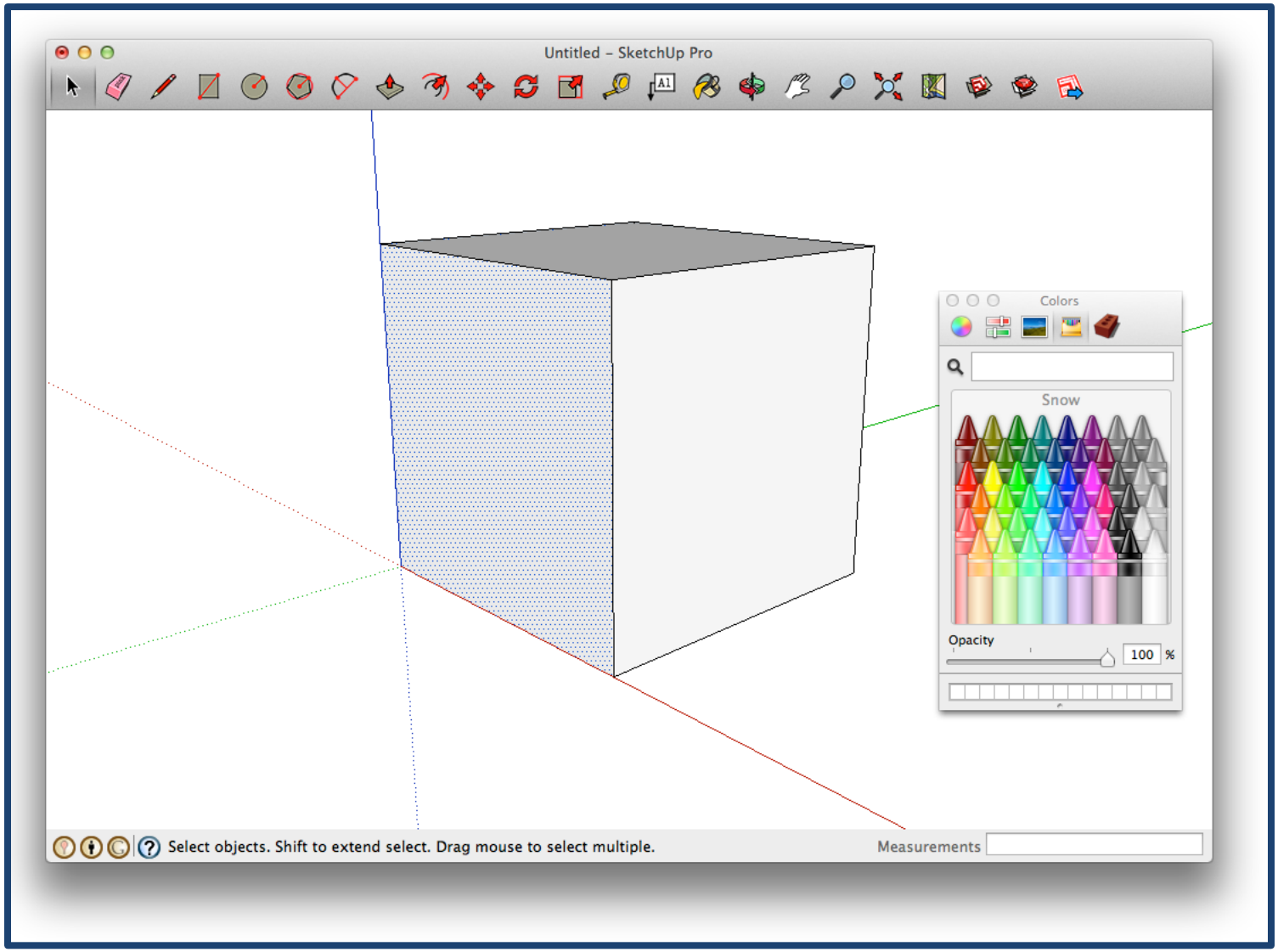
Just for example …



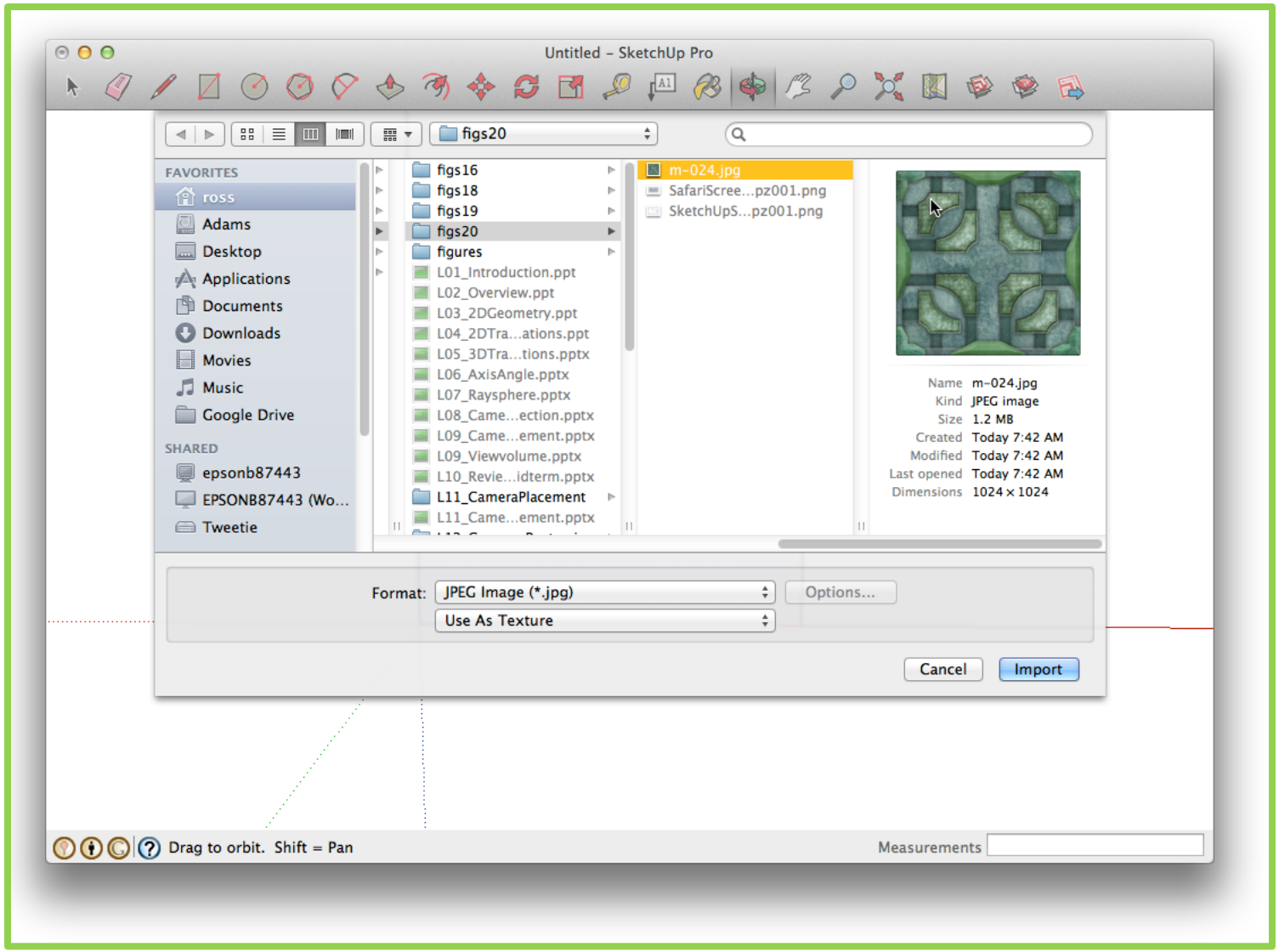
Texture Makers

- There are tons of them.. Some examples:
  - ftp://zdftp.zdnet.com/pub/private/sWIIB/graphics_multimedia_tools/graphics_tools/terltx32.zip
  - http://www.backgroundmagic.com/software/BGM.zip
  - http://www.metaworkshop.com/ftp/twinst.zip
  - http://216.156.212.112/photoseam.exe

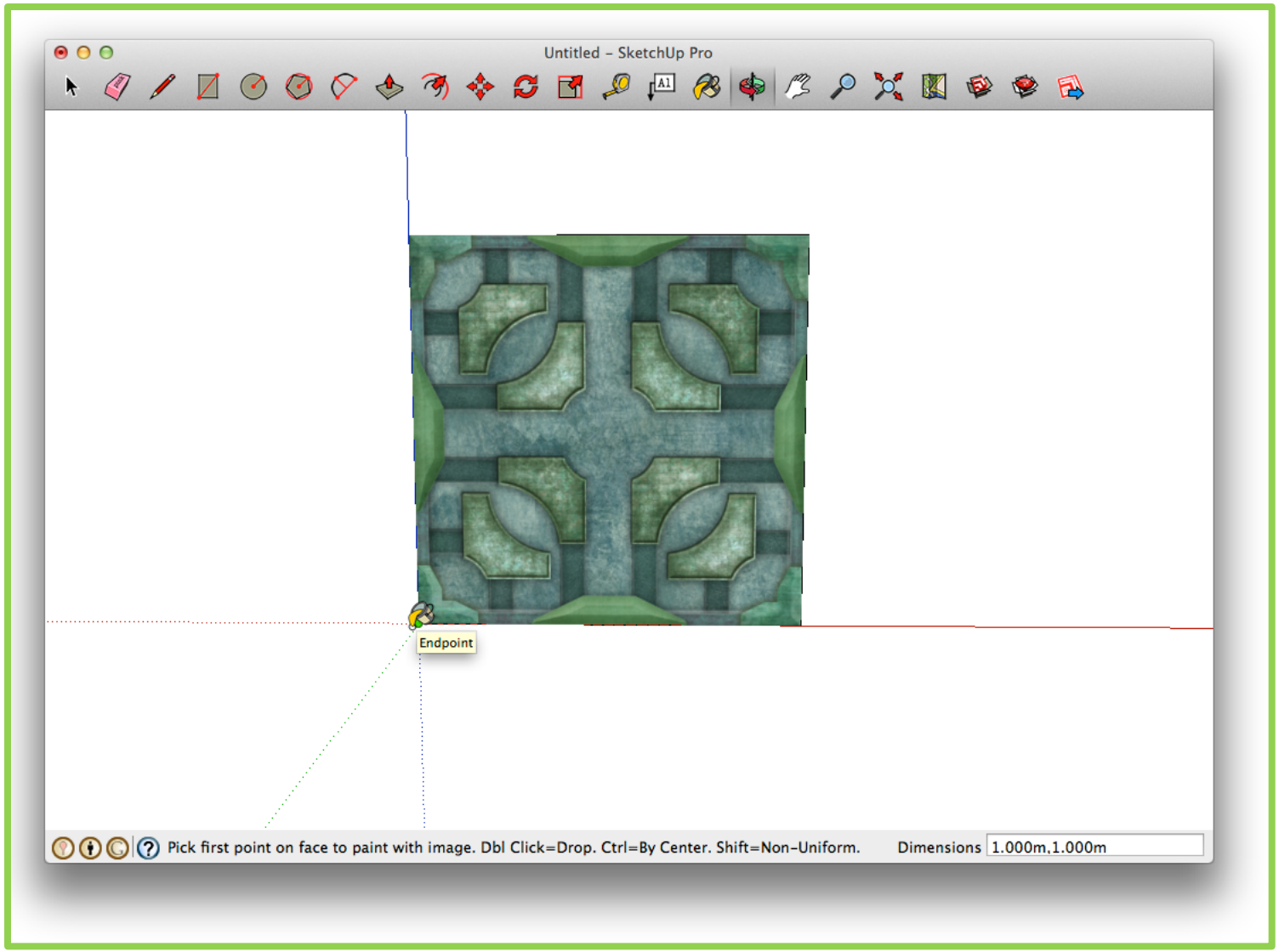
Example in SketchUp - Cube



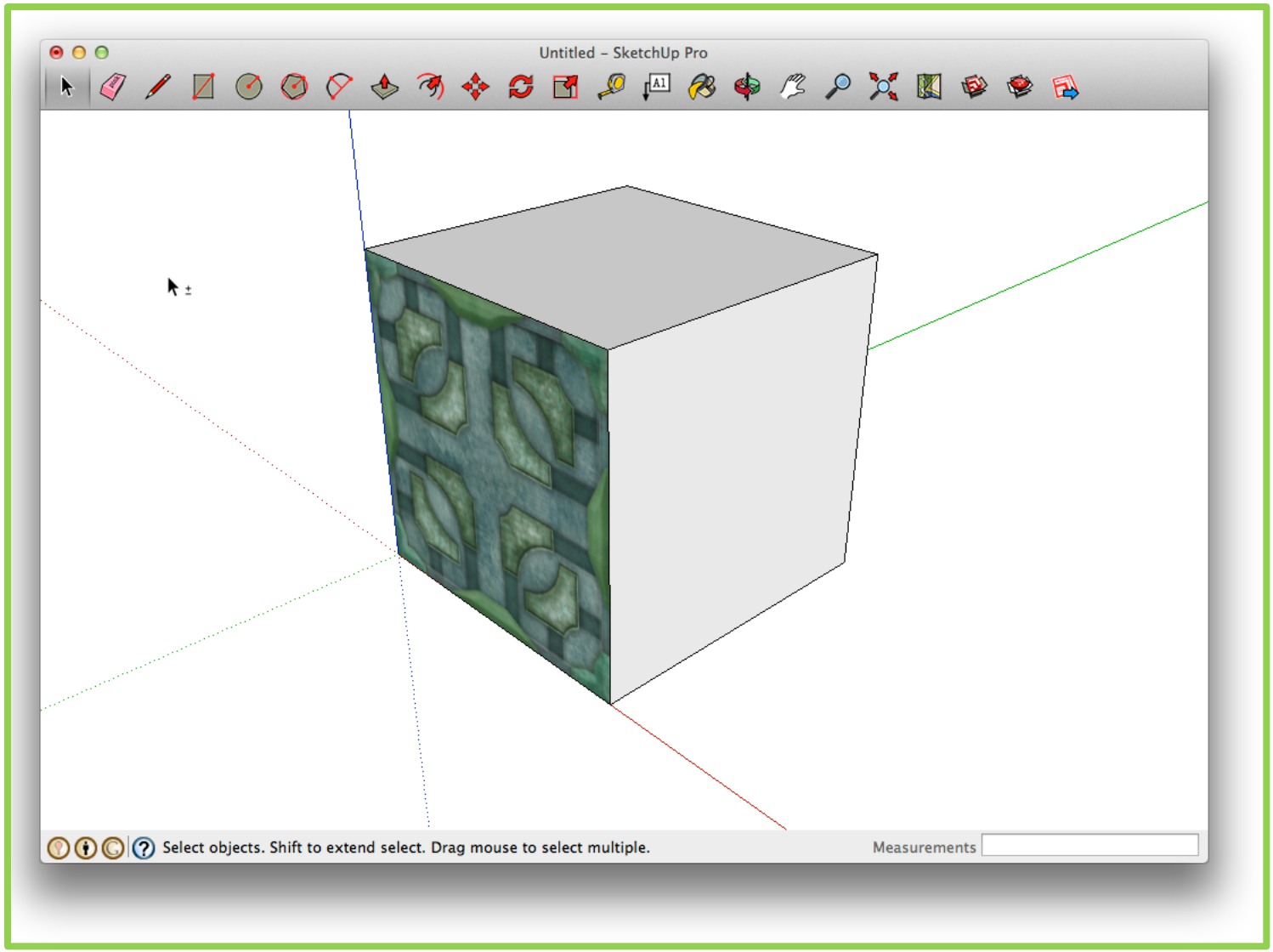
Import a Texture Image



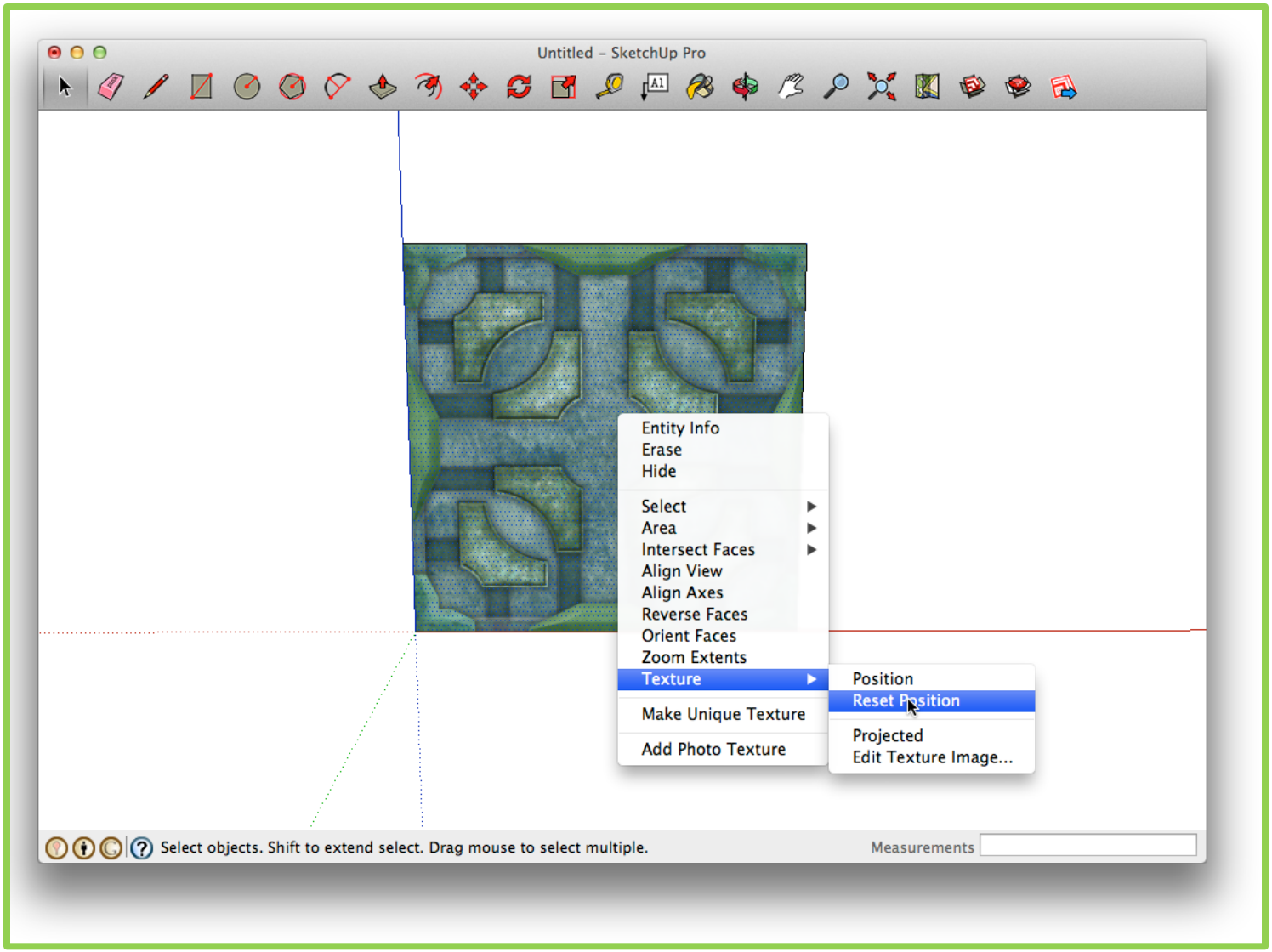
Place Texture on Face



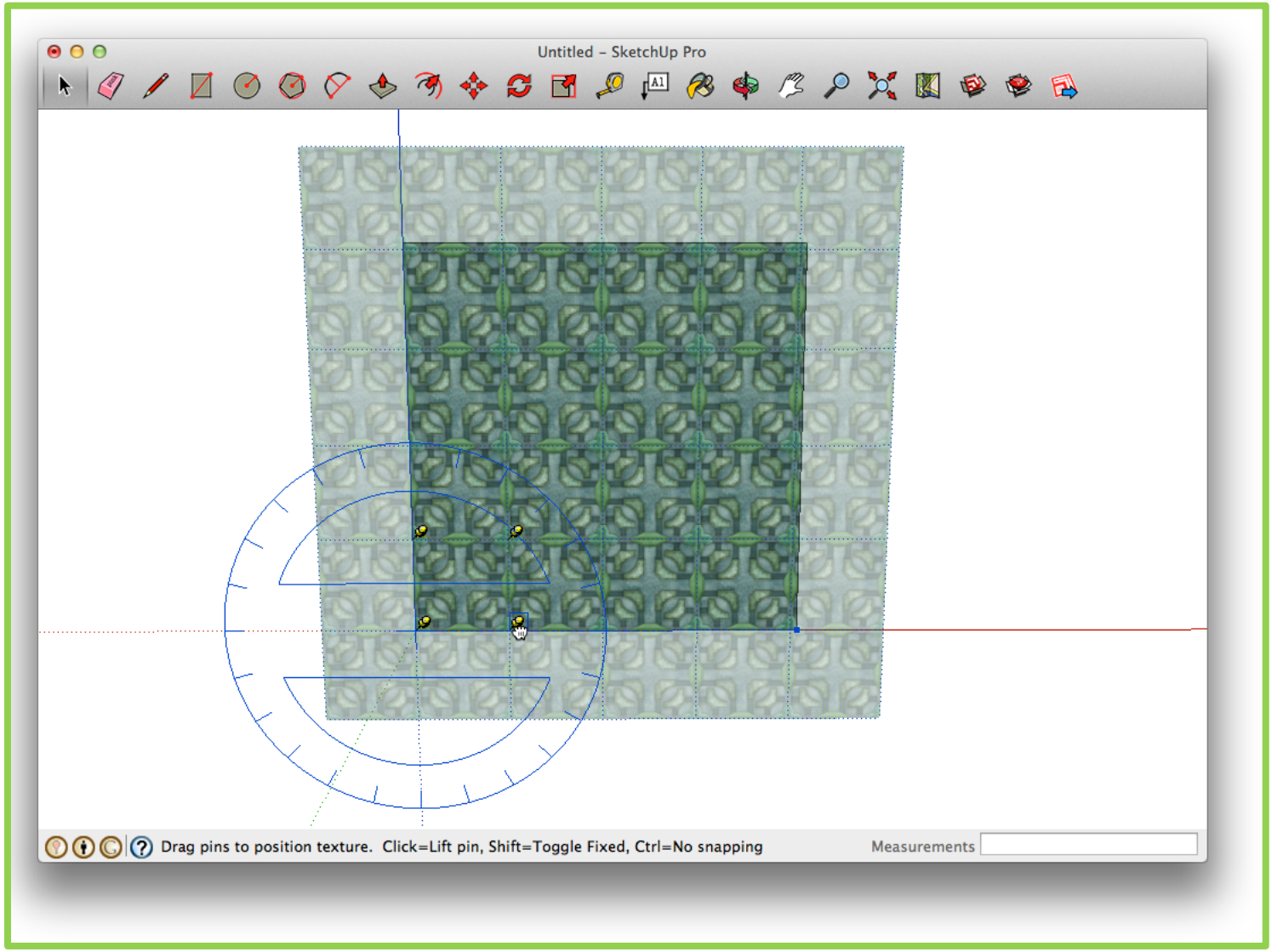
View the Result



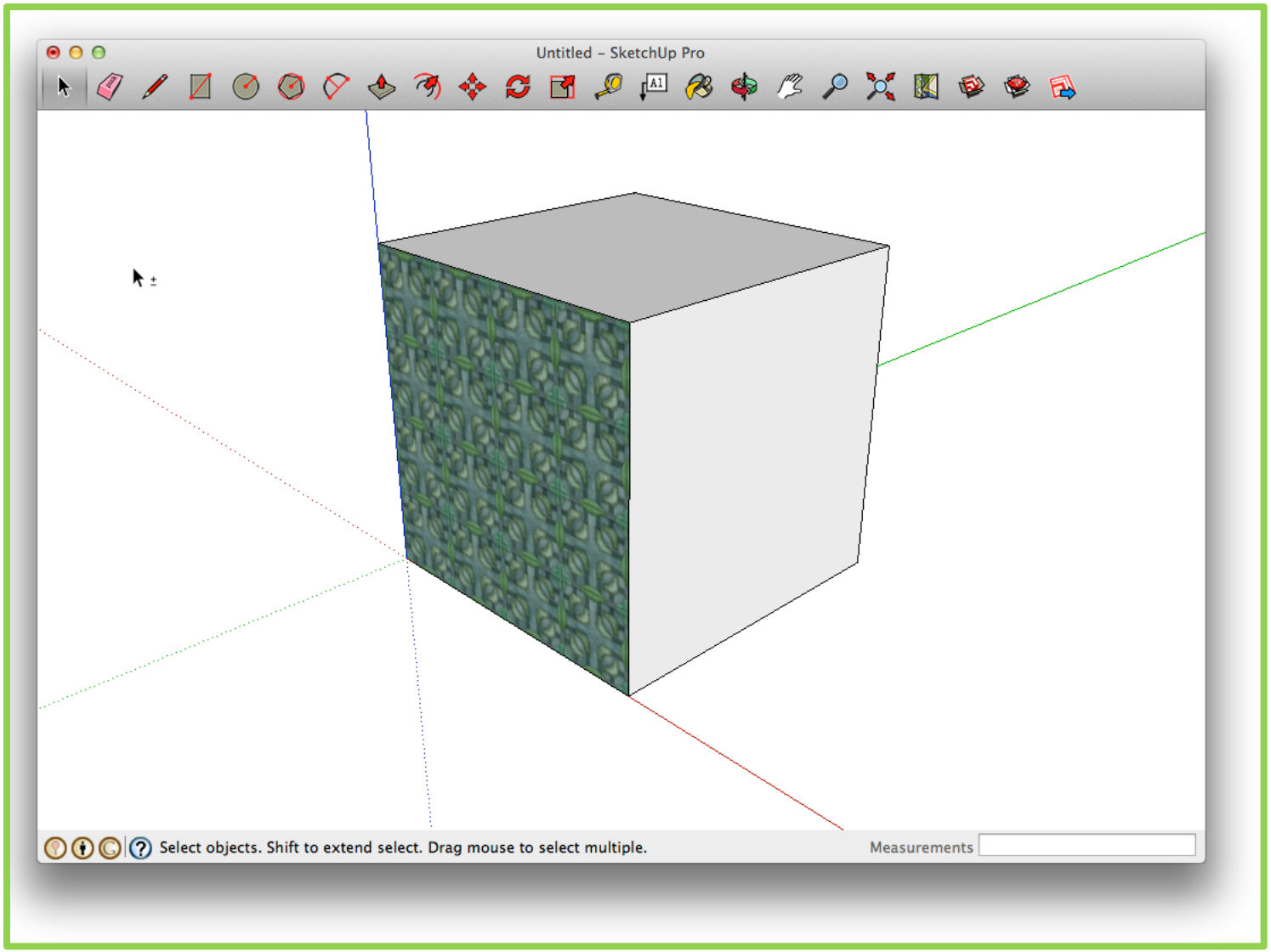
Position/Reset-Position



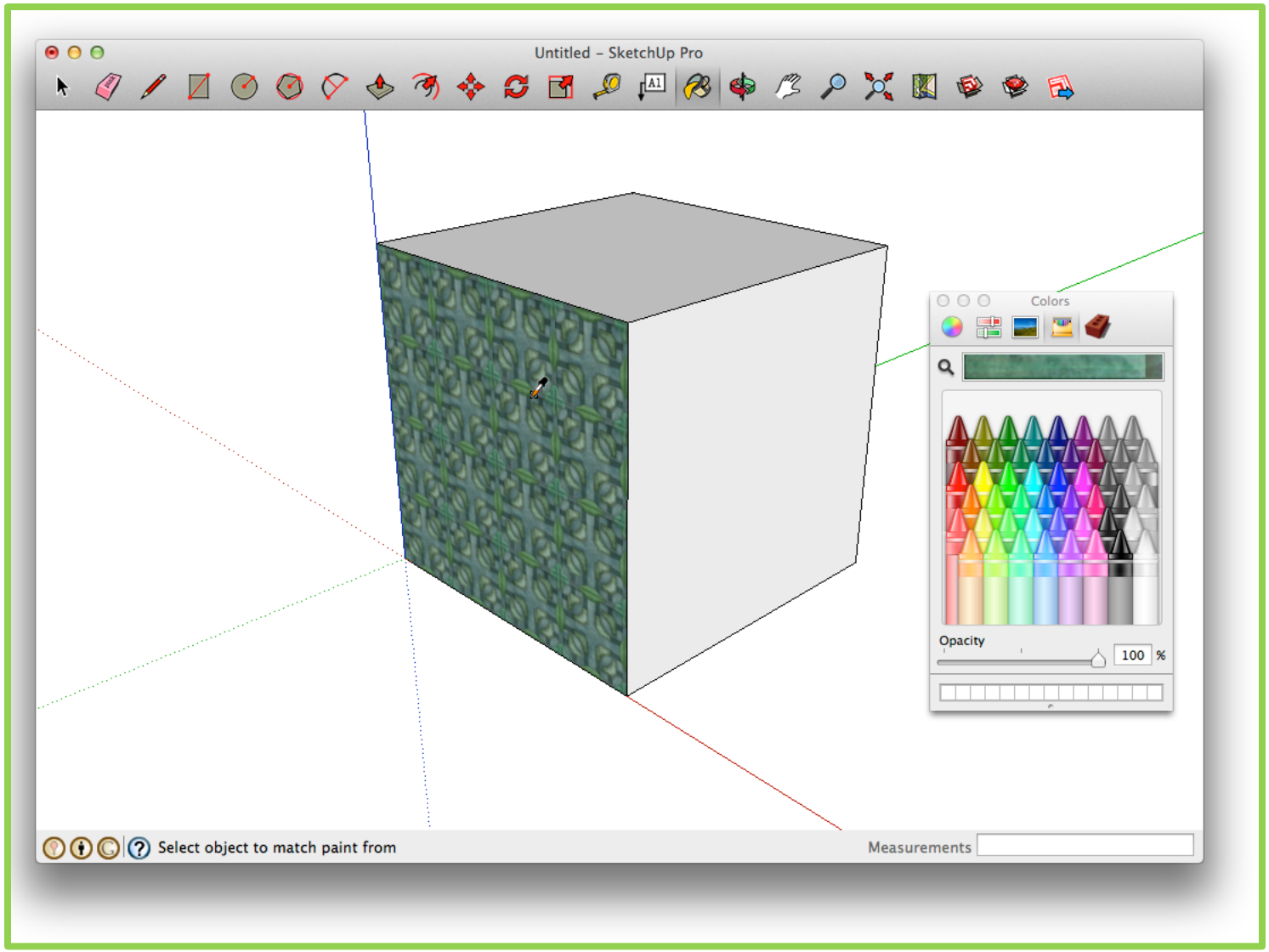
The Essence of Tiling



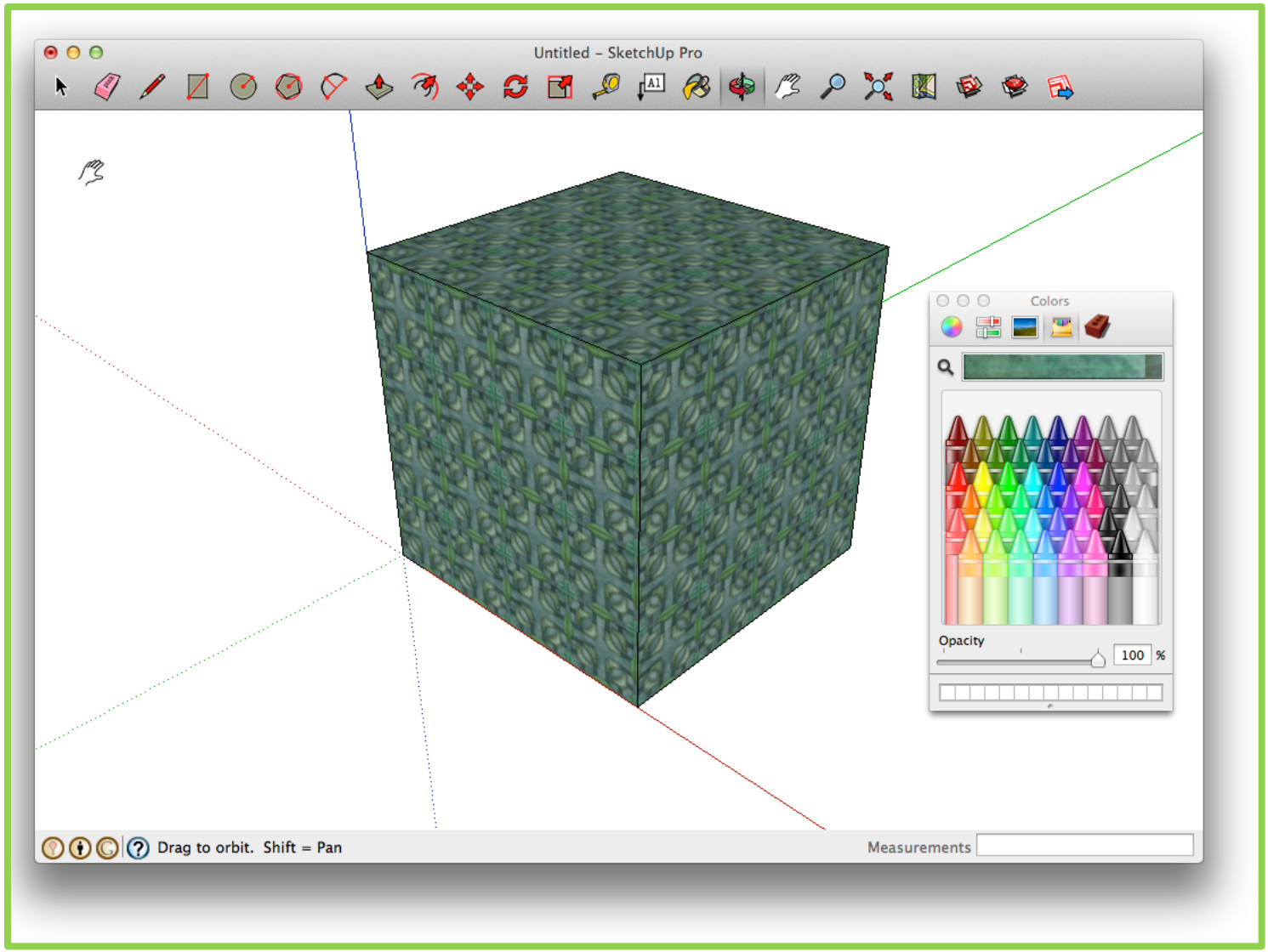
View the Result



Support for ‘Painting’ Textures

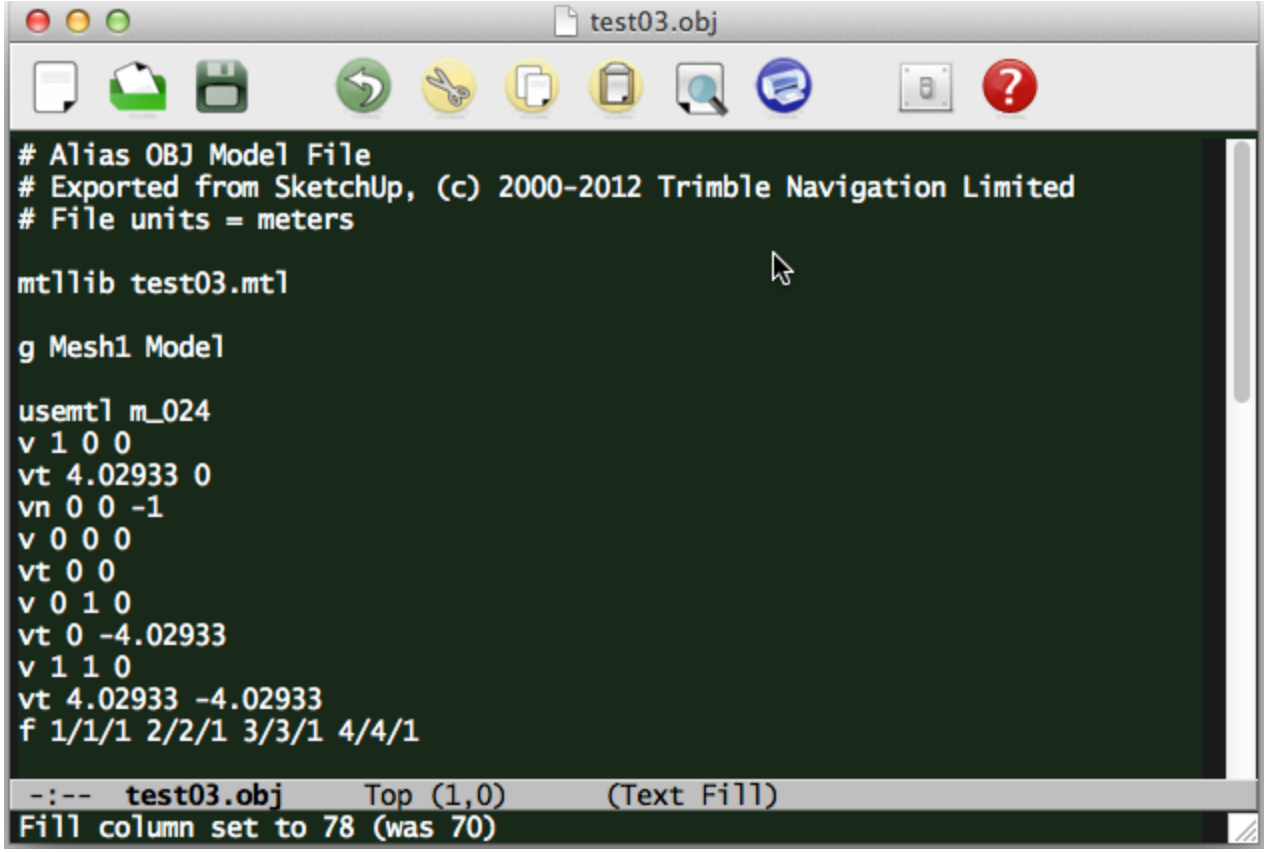


Final Result – Textured Cube

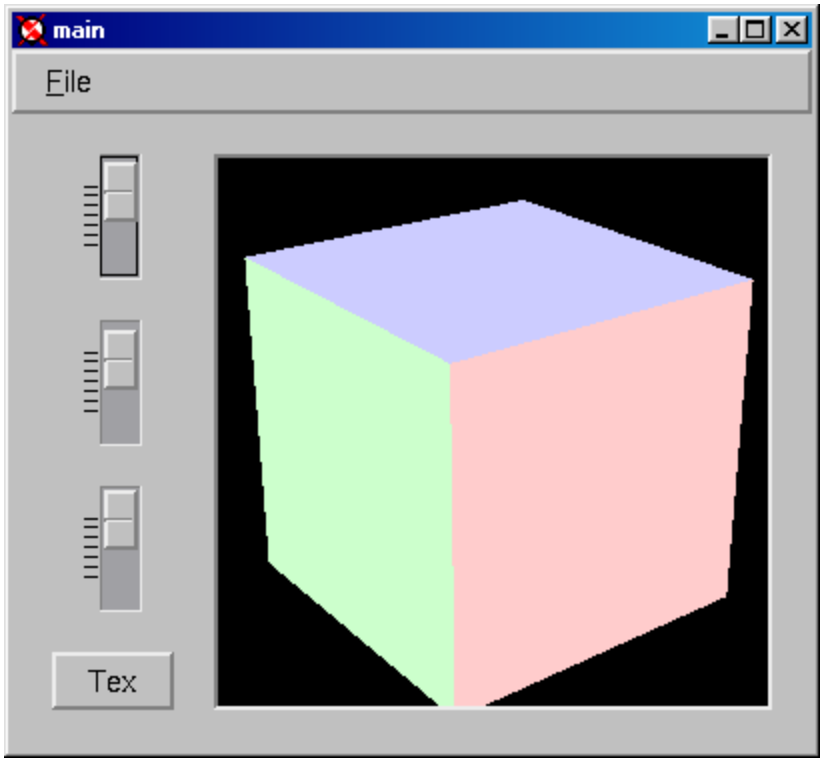


Alas - .obj support marginal

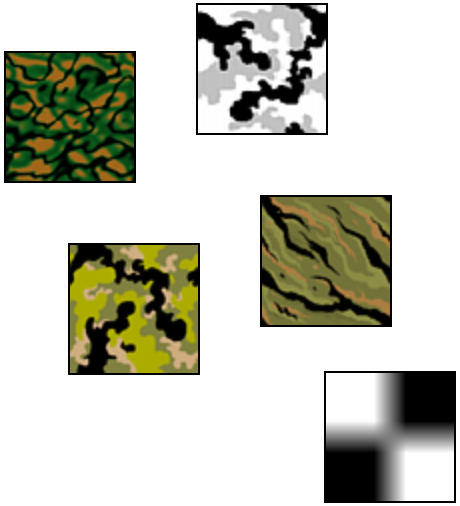
- Texture vertices do come through.
- File name also when using Blender.



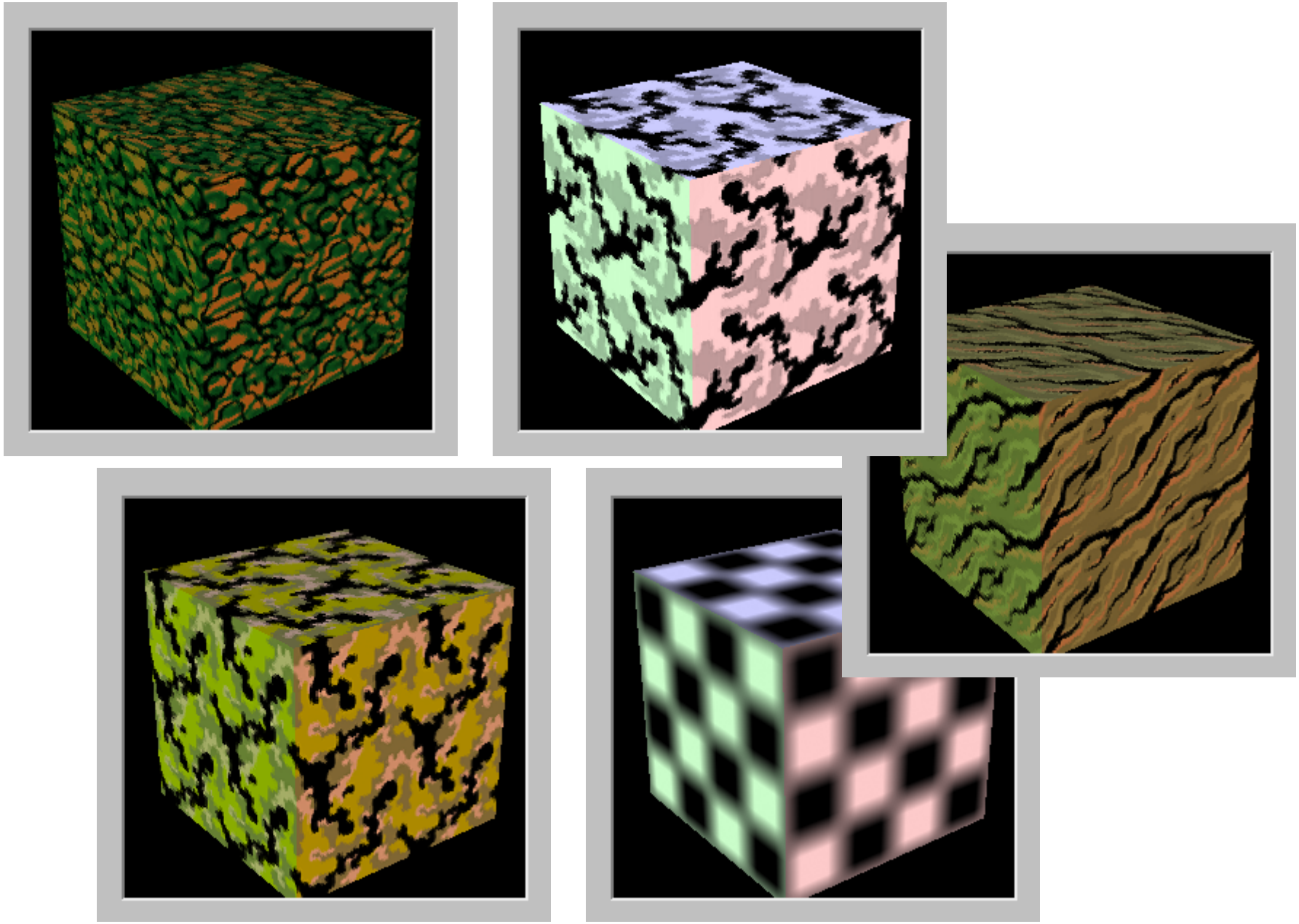
More Examples



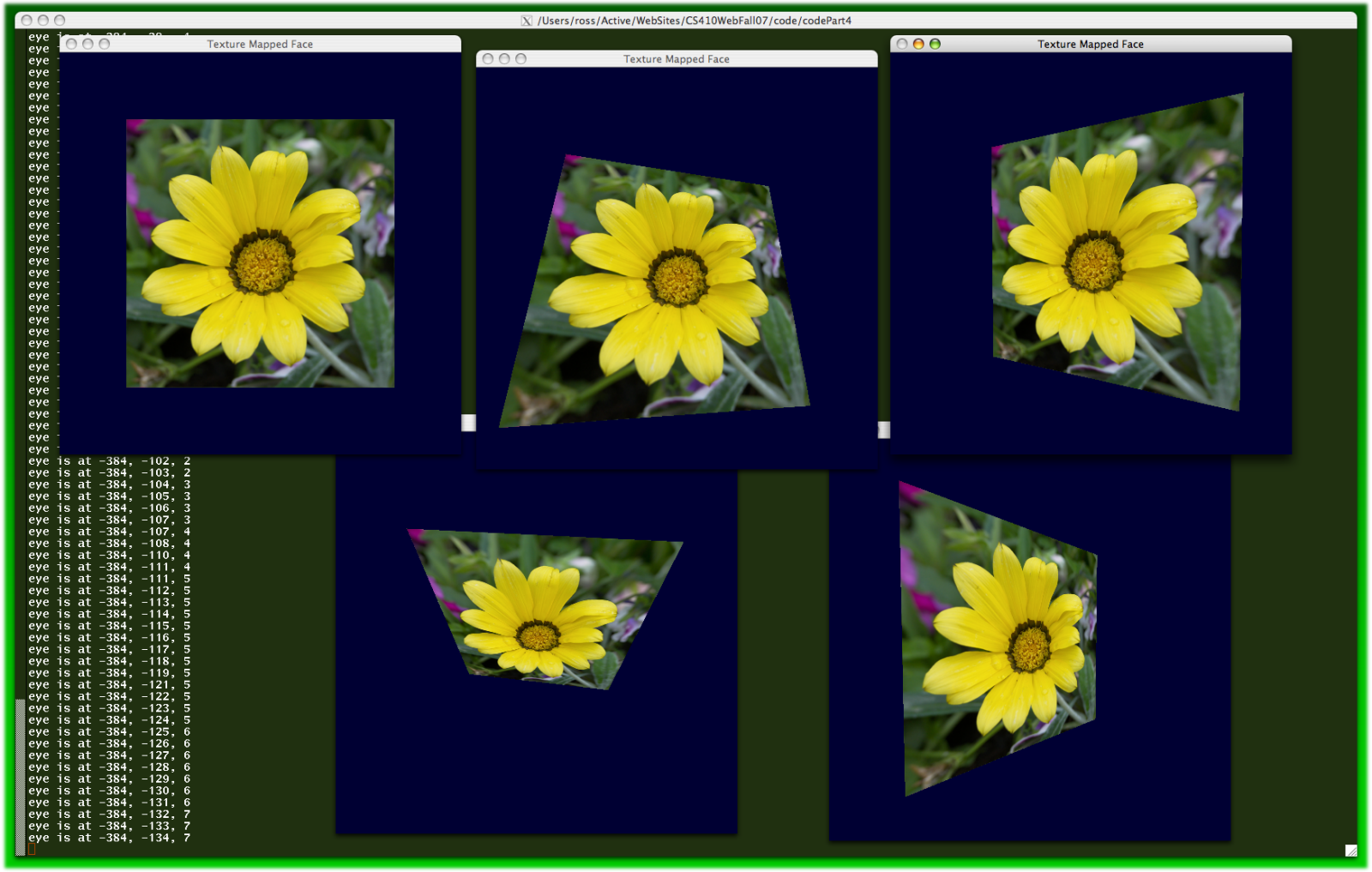
Surface



Textures



Texture Map from Image



A Touch of CS 510

- Consider the following 8 DOF transformation

$$\begin{bmatrix} u' \\ v' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$u = u'/w \ , \ v = v'/w$$

- The most practical way to specify an image transform is by providing 4 point correspondences

Computing Transformations

- Same basic idea we used for 3 points but bigger ….

$$\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2v_2 & -y_2v_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3v_3 & -y_3v_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4u_4 & -y_4u_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4v_4 & -y_4v_4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix}$$

Computing...

- So if we want the following mapping:

  $(0,0)\rightarrow(0,0)$, $(0,144)\rightarrow(0,144)$,

  $(152,0)\rightarrow(152,50)$, $(152,144)\rightarrow(152,94)$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 144 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 144 & 1 & 0 & -20736 \\ 152 & 0 & 1 & 0 & 0 & 0 & -23104 & 0 \\ 0 & 0 & 0 & 152 & 0 & 1 & -7600 & 0 \\ 152 & 144 & 1 & 0 & 0 & 0 & -23104 & -21888 \\ 0 & 0 & 0 & 152 & 144 & 1 & -14288 & -13536 \end{bmatrix}$$

…More Computing...

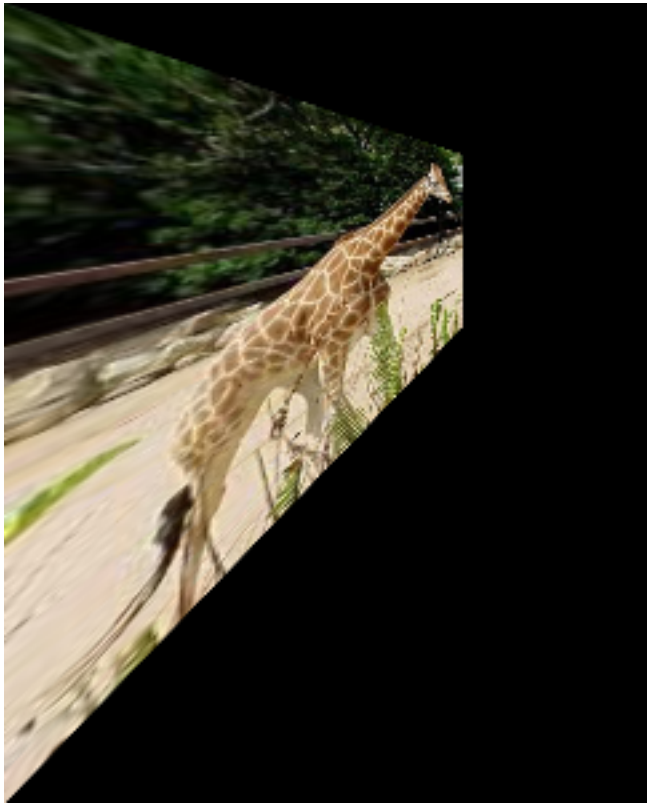
What does This say about x?

$$\underbrace{\begin{bmatrix} -.014 & -.023 & .007 & .023 & .014 & .022 & -.007 & -.022 \\ -.007 & 0 & .007 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -.002 & -.014 & .002 & .007 & .002 & .014 & -.002 & -.007 \\ -.007 & -.007 & .007 & .007 & .007 & 0 & -.007 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ .000 & -.000 & .000 & .000 & .000 & .000 & -.000 & .000 \\ -.000 & 0 & .000 & 0 & .000 & 0 & -.000 & 0 \end{bmatrix}}_{M^{-1}} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 144 \\ 152 \\ 50 \\ 152 \\ 94 \end{bmatrix}}_{\text{u\&v vector}} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} 3.274 \\ 0 \\ 0 \\ 1.077 \\ 1 \\ 0 \\ .01497 \\ 0 \end{bmatrix}$$

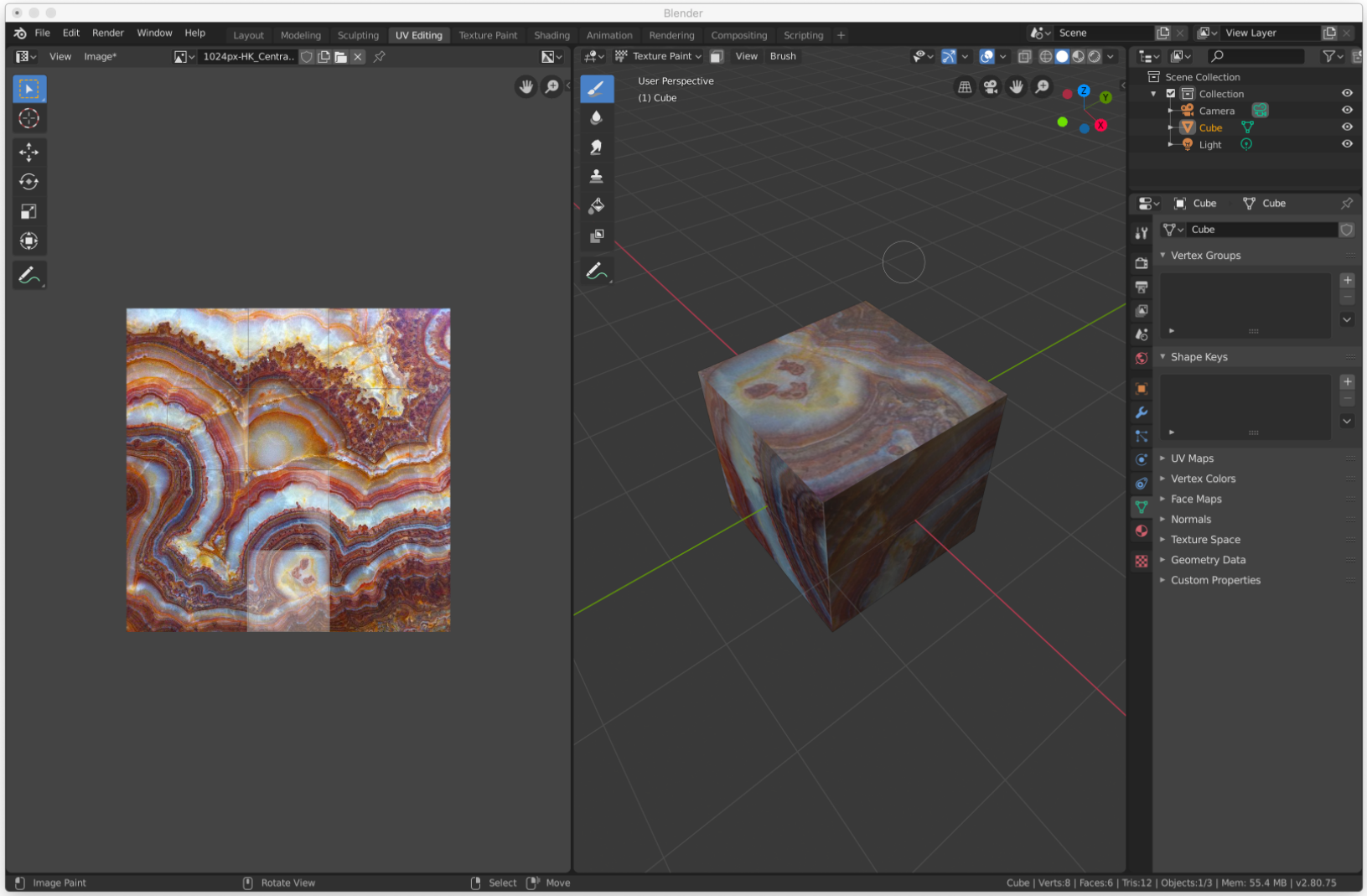
Remember the earlier WLOG? $c = u_1, \ldots$

How does This alter it?

…yields $\begin{bmatrix} 3.274 & 0 & 0 \\ 1.077 & 1 & 0 \\ .01497 & 0 & 1 \end{bmatrix}$



Texture in Blender



Credit where credit is due – YouTube video:
"Blender 2.8 Beginner Textures and Materials Tutorial"