CS193P - Lecture 19

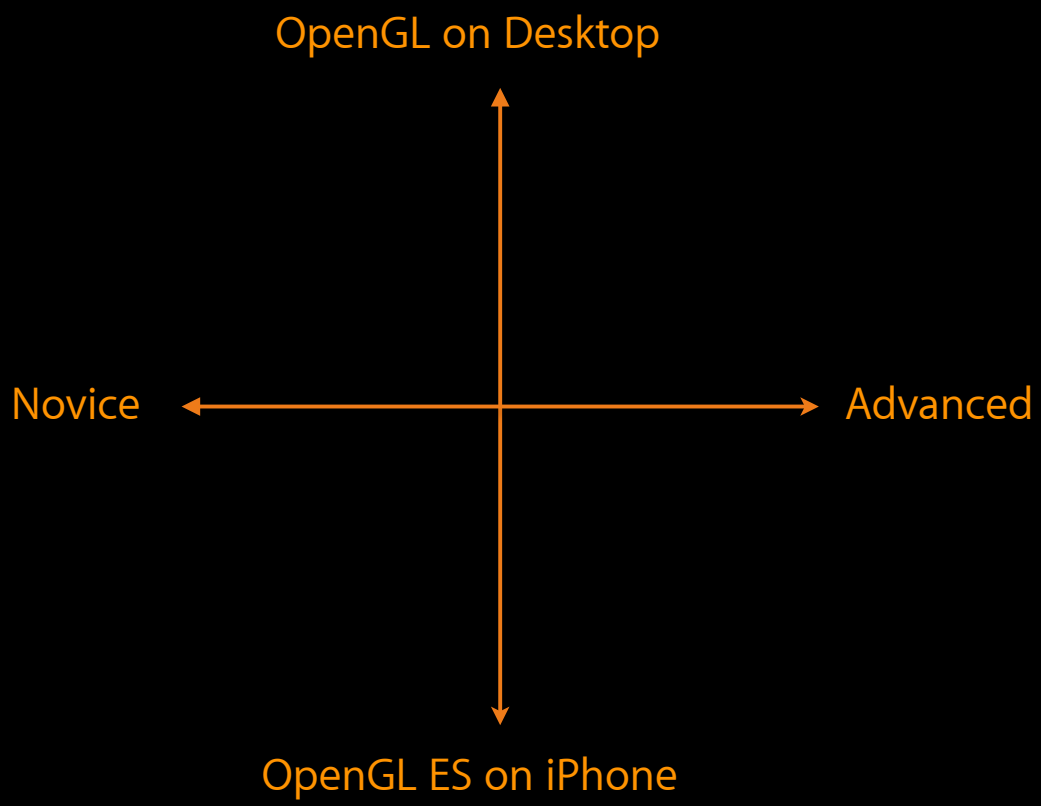
iPhone Application Development

OpenGL ES

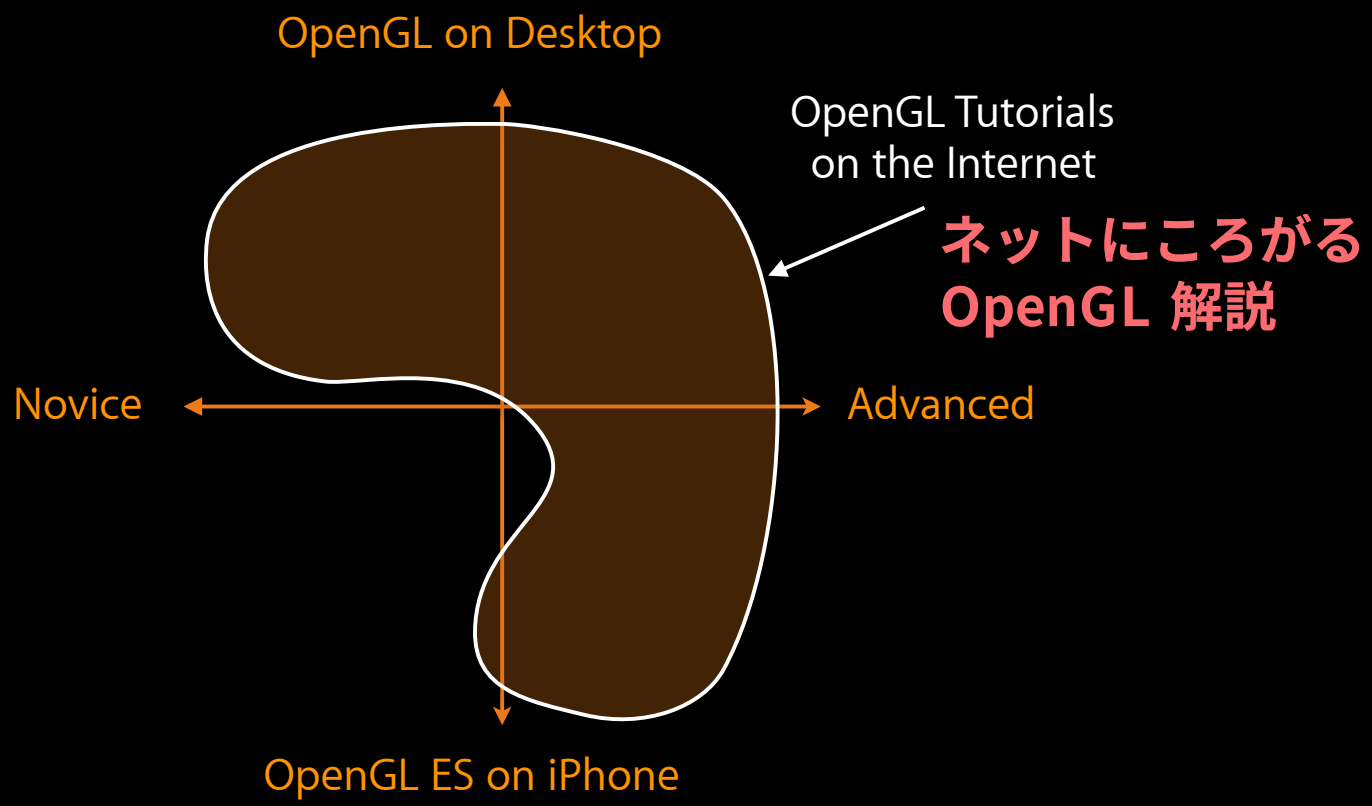
Announcements

- **Final projects** due in 7 days
  - Tuesday, March 16th 11:59 pm
  - Submit:
    - Code
    - Power-point/Keynote slides
    - ReadMe file
- Final project **demos**
  - March 18, from 12:15 - 3:15 pm in Hewlett 201
  - 2 minute presentation, followed by demo-fair
    - Rapid-fire!!
    - Time limit strictly enforced
  - Let us know if you do not want to be recorded

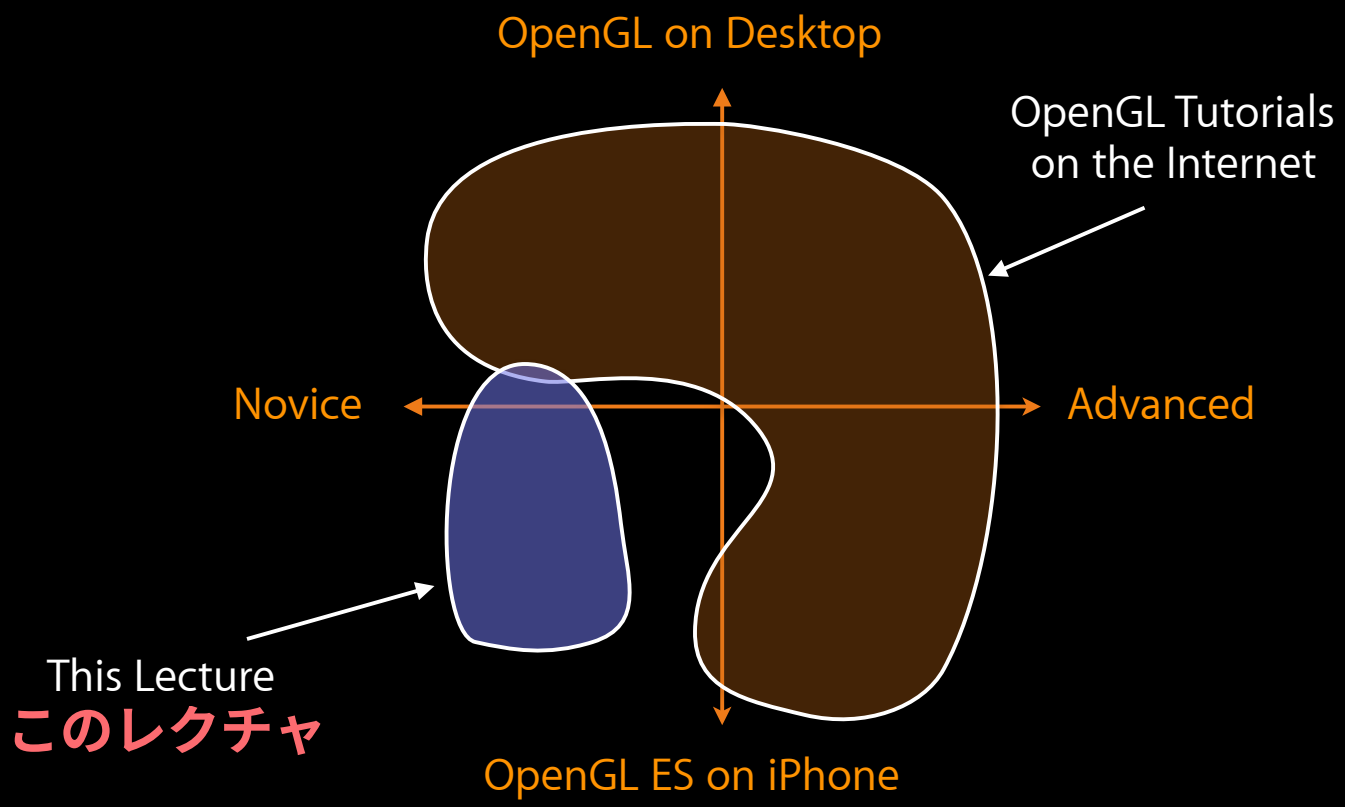
Today's Topics 今日のトピックス



Today's Topics



Today's Topics



Today's Topics

- OpenGL Overview　**OpenGL の概要**
- Coordinate Systems and Transformations　**座標系と変換**
- Drawing Geometry　**図形の描画**
- Using Textures　**テクスチャの貼り付け**
- Other Details　**その他もろもろ**

OpenGL Overview

OpenGL Overview OpenGL の概要

OpenGL Overview

- Software interface for graphics hardware
  グラフィクスハードウェア（GPU）へのインタフェース

OpenGL Overview

- Software interface for graphics hardware
- Quickly render 2D or 3D graphics

  ２D・３Dグラフィクスの高速描画

OpenGL Overview

- Software interface for graphics hardware
- Quickly render 2D or 3D graphics
- Hardware implementation agnostic

  ハードウェア実装を「見えなくする」

OpenGL Overview

- Software interface for graphics hardware
- Quickly render 2D or 3D graphics
- Hardware implementation agnostic

OpenGL Overview

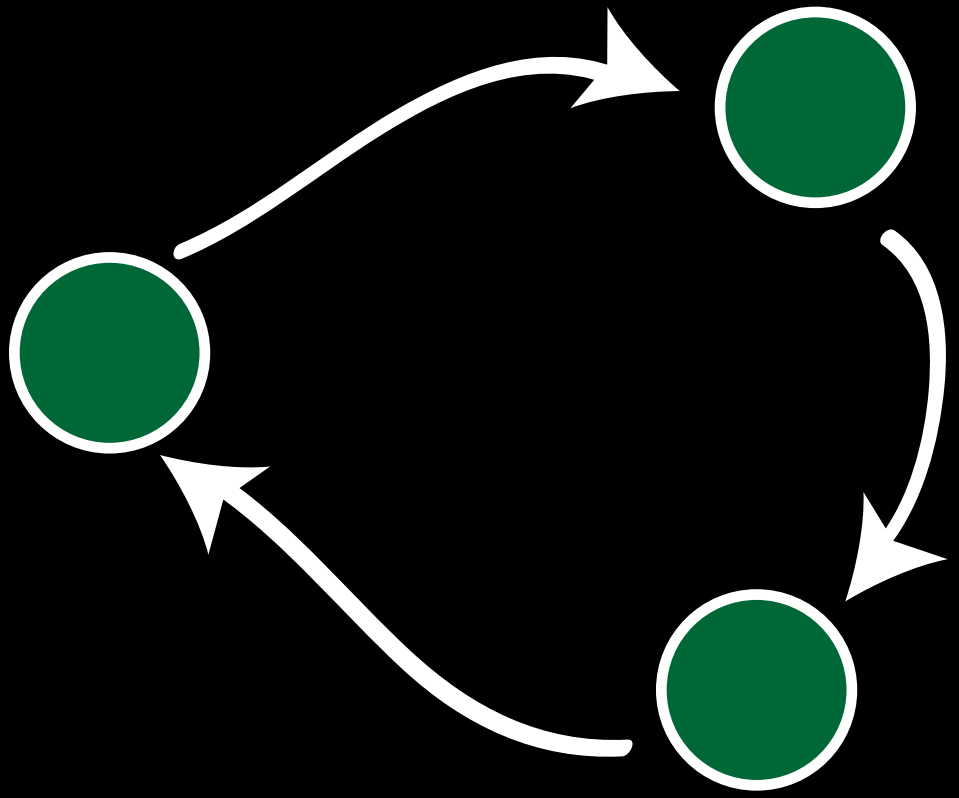
- Software interface for graphics hardware
- Quickly render 2D or 3D graphics
- Hardware implementation agnostic

- OpenGL ES is a subset of OpenGL
  - GLUT and GLU are not available on the iPhone

OpenGL ES は OpenGL のサブセット
GLUT や GLU は使えない

OpenGL is a state machine

OpenGL は状態マシン



内部に状態変数があり，
コマンドを実行するごとに
状態変数の内容が変わっていく

だから，同じコマンドでも，
状態に応じて実行内容が変わる

OpenGL is a state machine

- Change Machine State 状態を変化させる

```
glEnable(); glDisable(); glMatrixMode(); glBindFramebufferOES();
glViewport(); glVertexPointer(); glColorPointer(); glTranslatef() ...
```

- Issue Drawing Commands 描画コマンドの発行

```
glDrawArrays(); glDrawElements(); ...
```

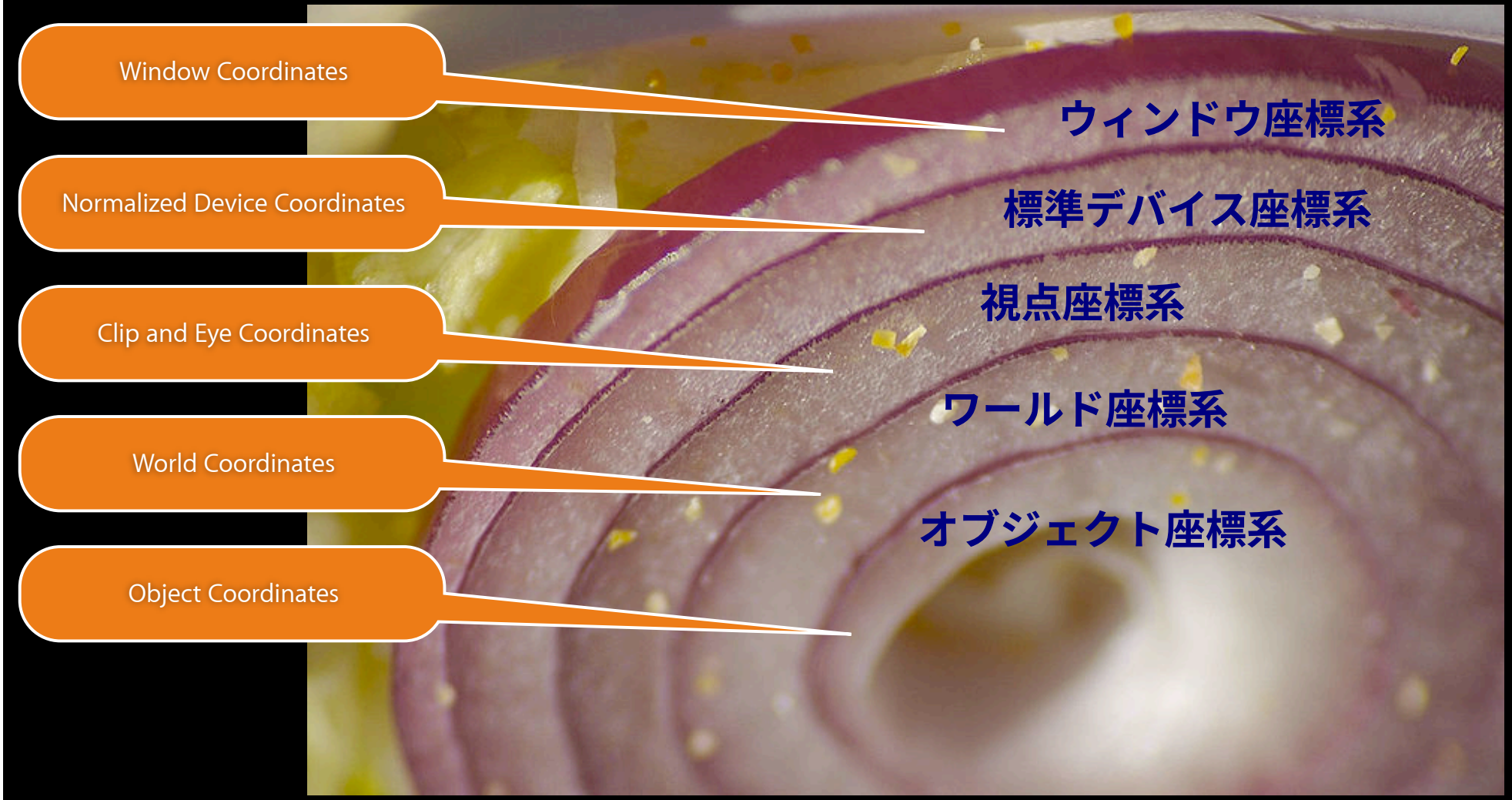
- Read Back State, Drawing Results 内部状態や結果の読み出し

```
glGetBooleanv(); glGetFloatv(); glReadPixels(); ...
```

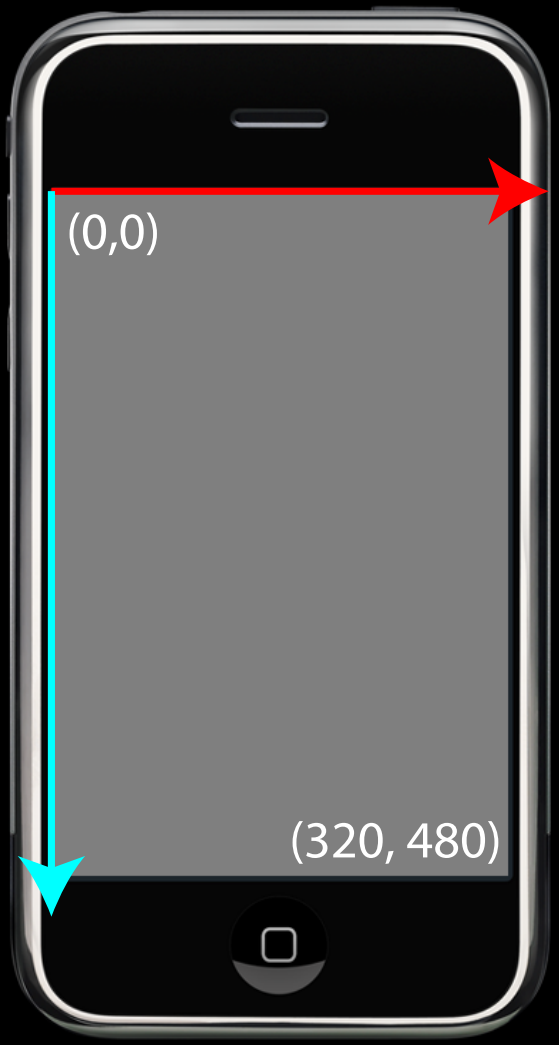
Coordinate Systems

座標系

The Coordinate System Onion

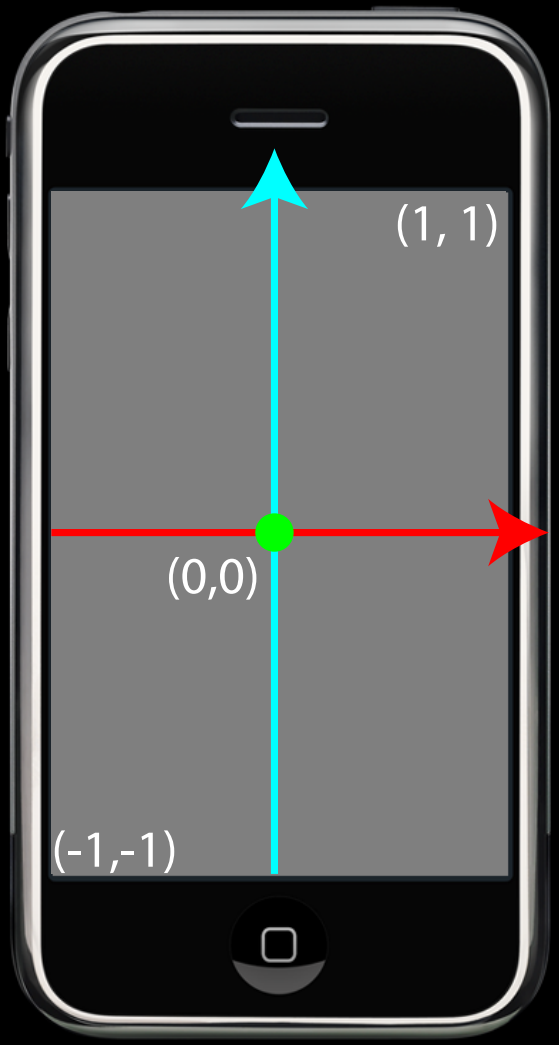


Window Coordinates ウィンドウ座標系

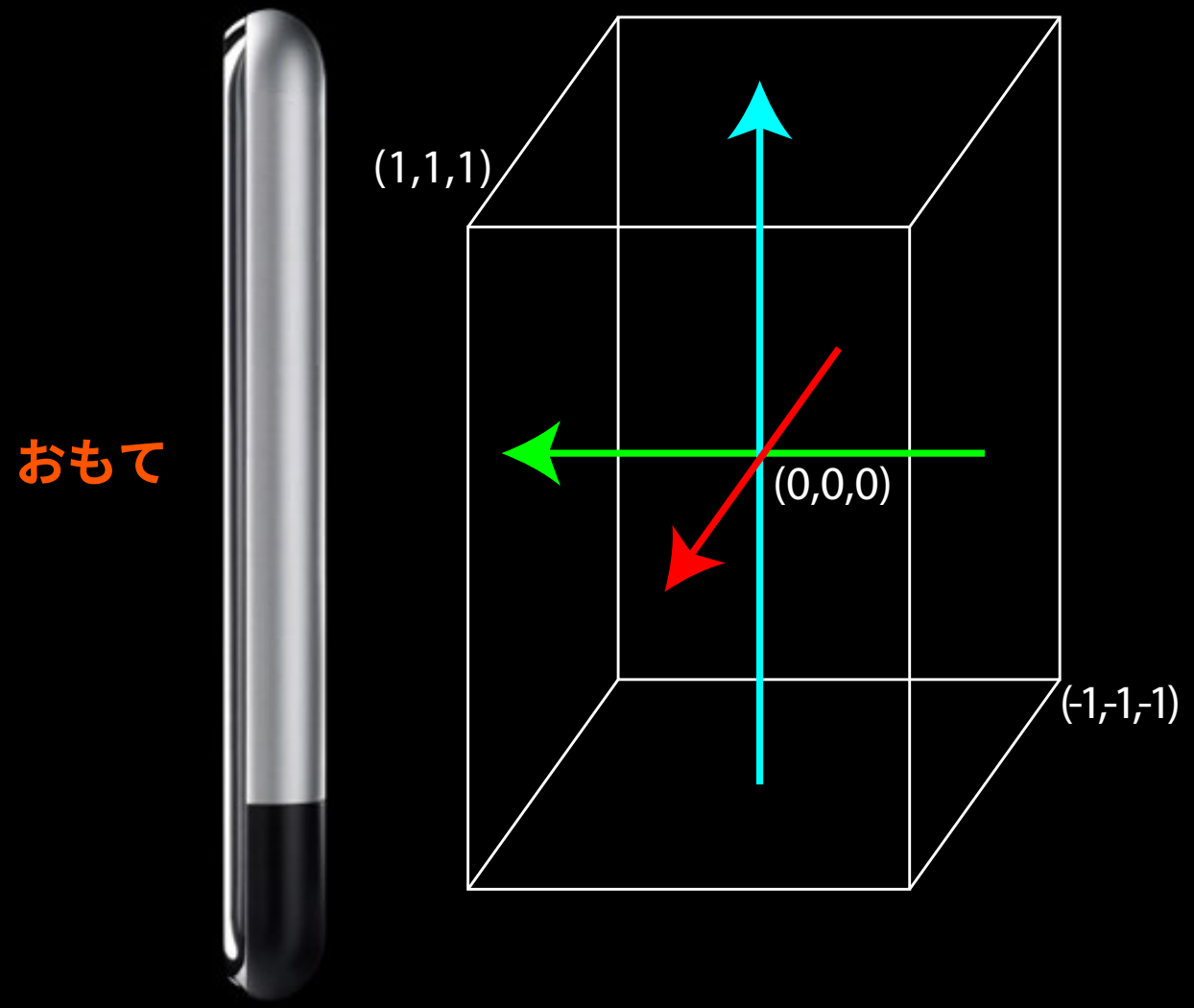


Normalized Device Coordinates

標準デバイス座標系



Normalized Device Coordinates



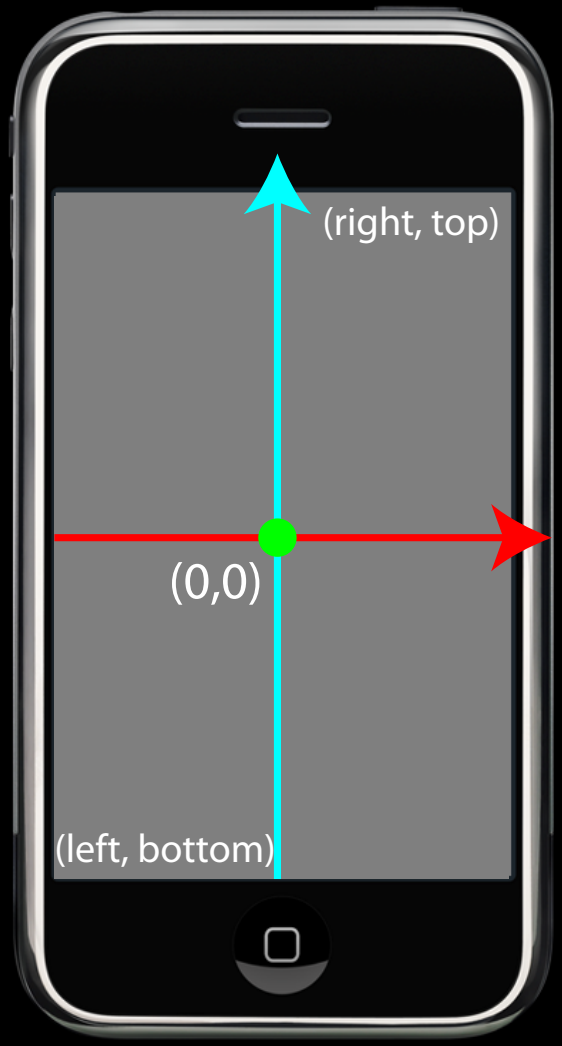
Clip Coordinates

ILLUSTRATION
NOT FOUND

なんだこりゃ

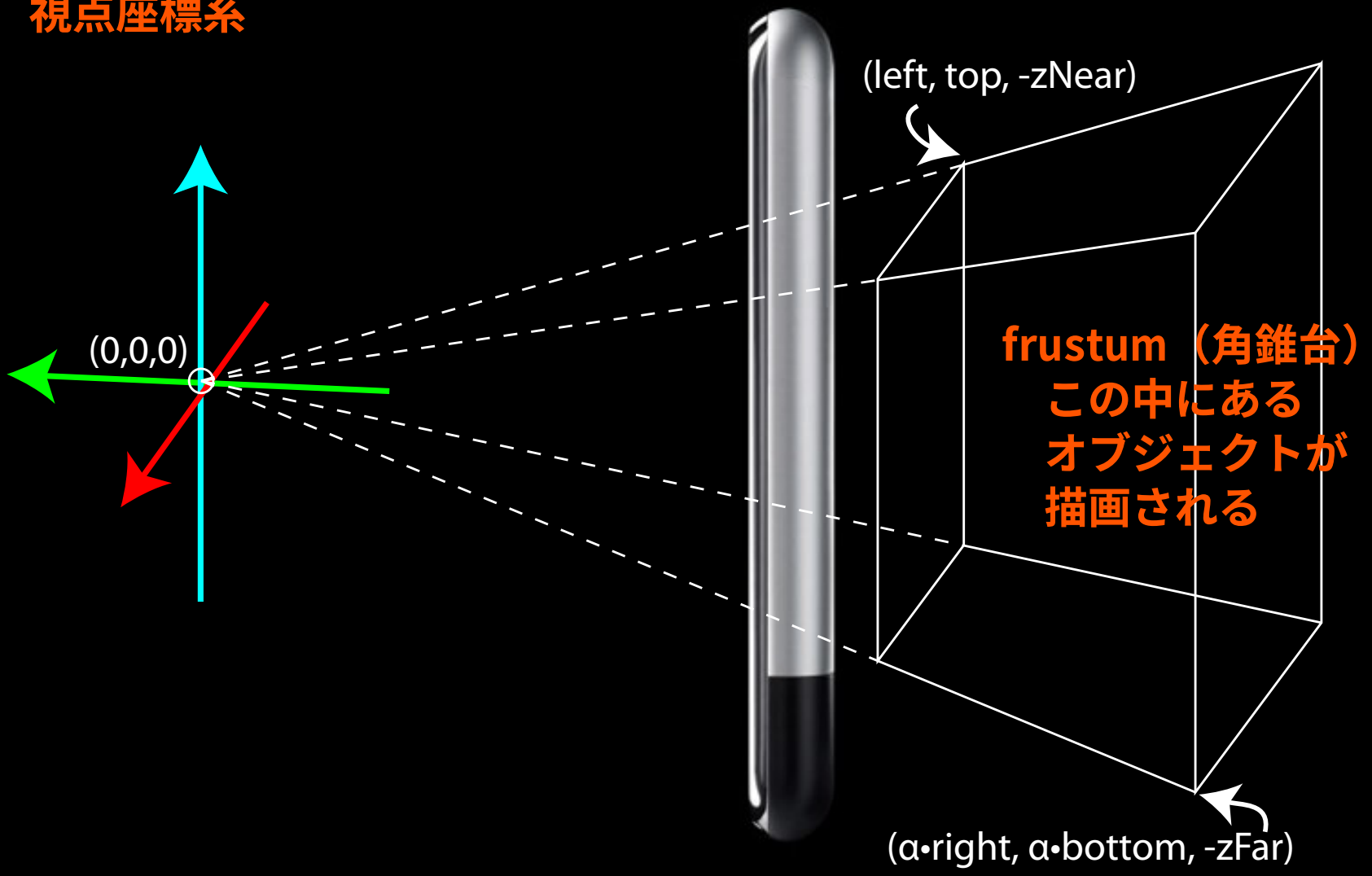
Eye Coordinates

視点座標系



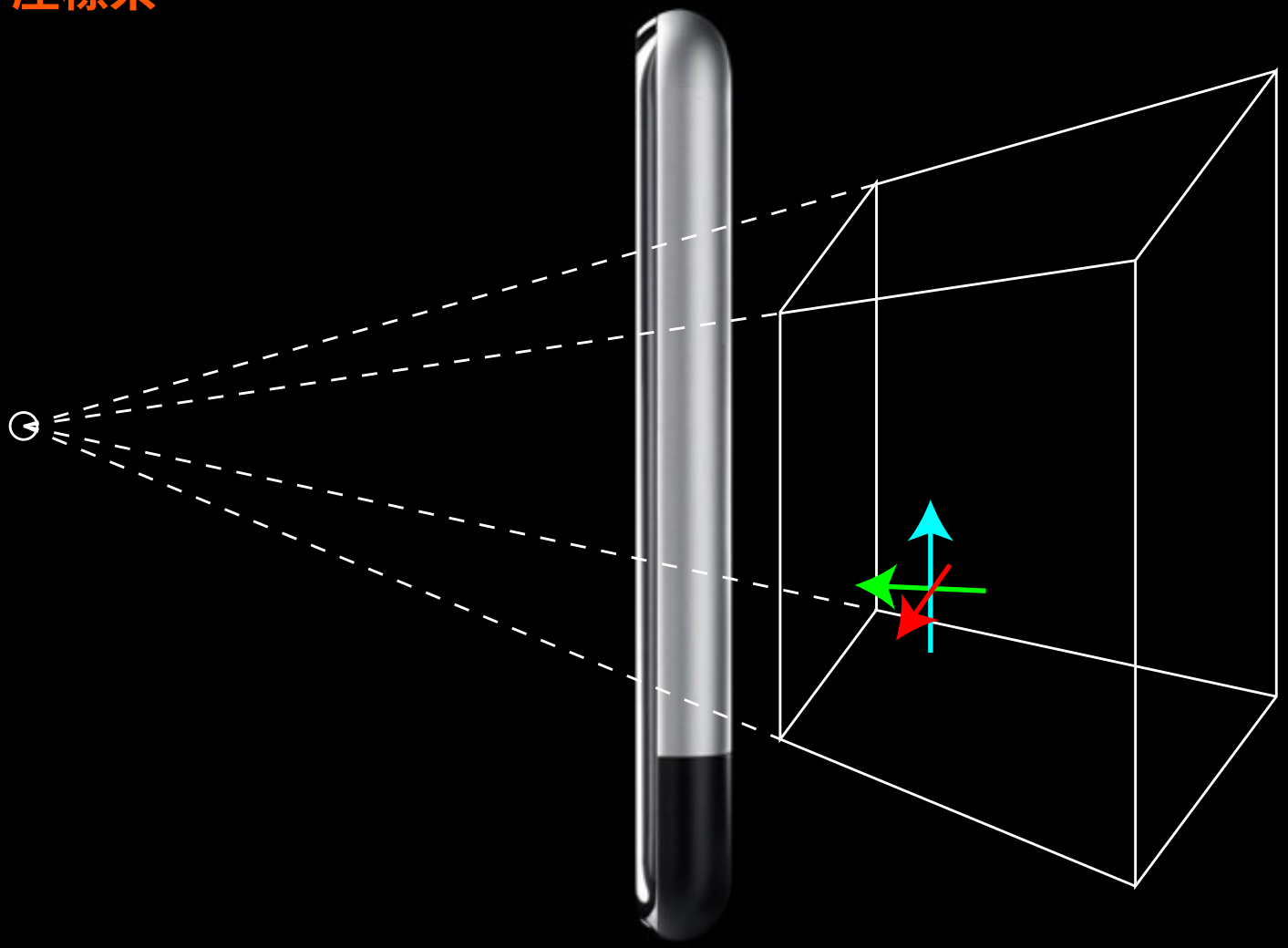
Eye Coordinates

視点座標系



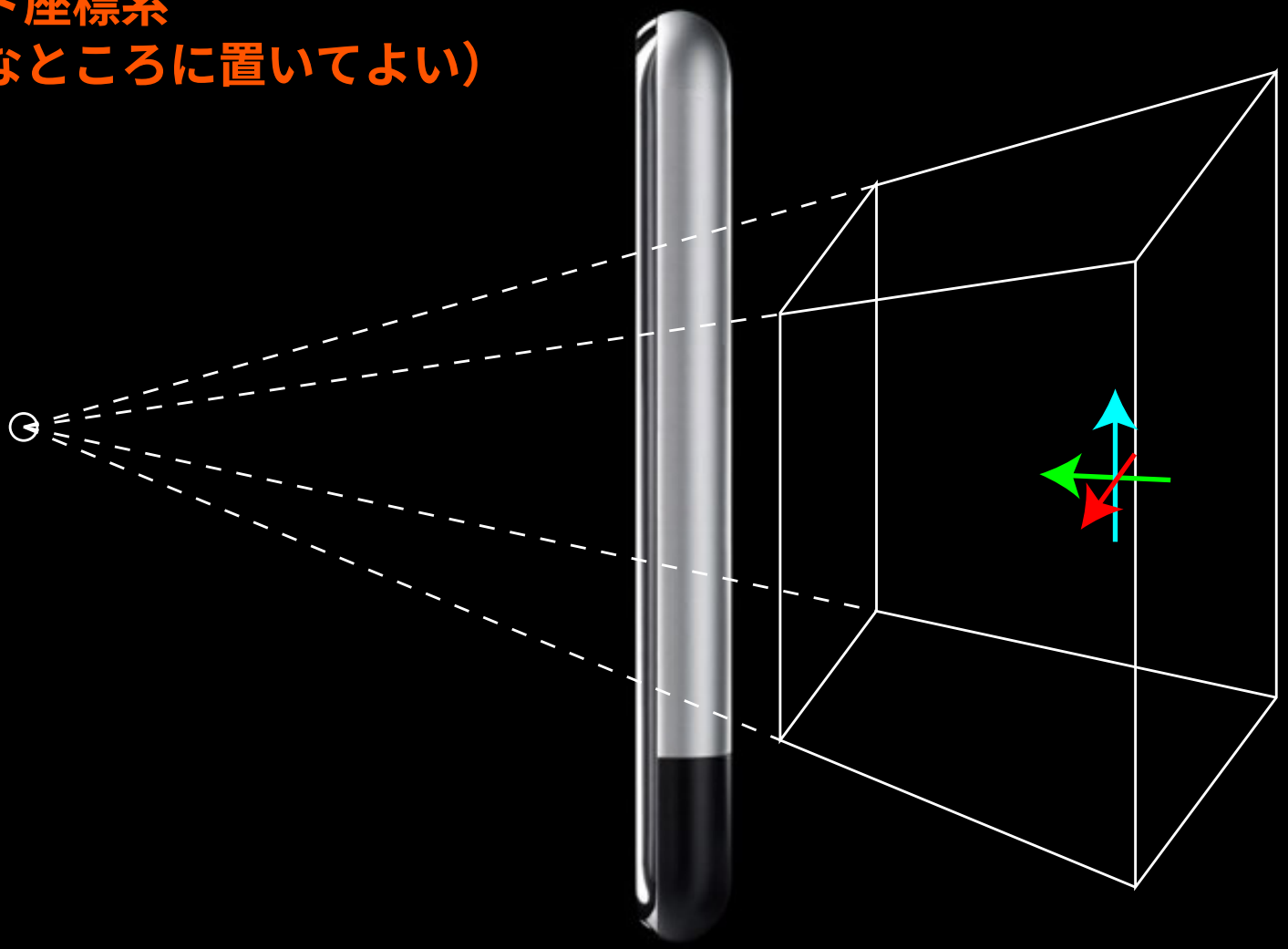
World Coordinates

ワールド座標系



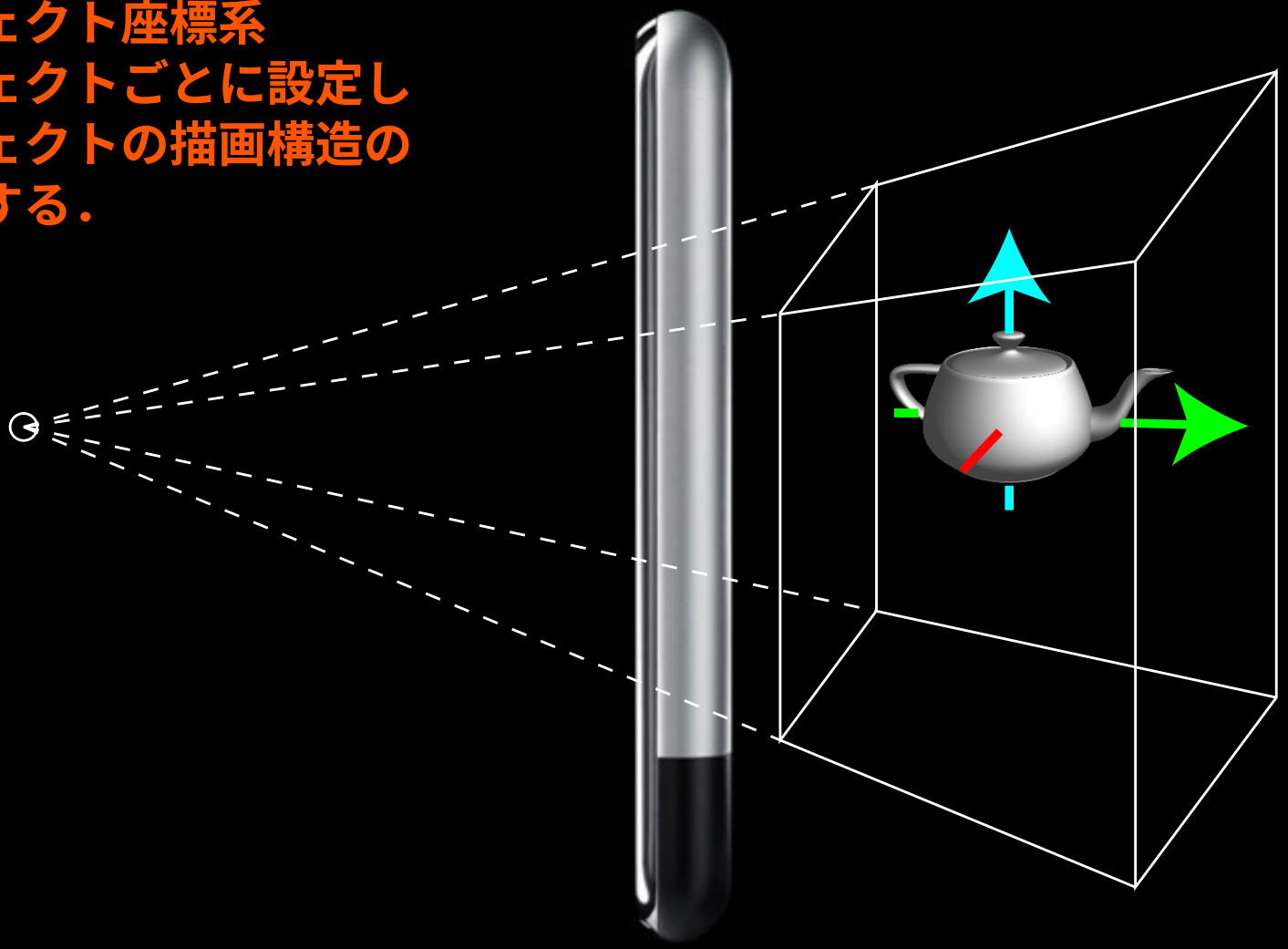
World Coordinates

ワールド座標系
（好きなところに置いてよい）



Object Coordinates

オブジェクト座標系
オブジェクトごとに設定し
オブジェクトの描画構造の
基準とする．



Demo

Transformations

Transformations Cheat Sheet

変換（カンペ）

```
glMatrixMode(GL_PROJECTION);
glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

glOrthof(left, right, bottom, top, zNear, zFar);
glFrustumf(left, right, bottom, top, zNear, zFar);

glRotatef(degrees, x, y, z);
glTranslatef(x, y, z);
glScalef(x, y, z);
glMultMatrixf(matrix);

glPushMatrix();
glPopMatrix();
```

Drawing Geometry

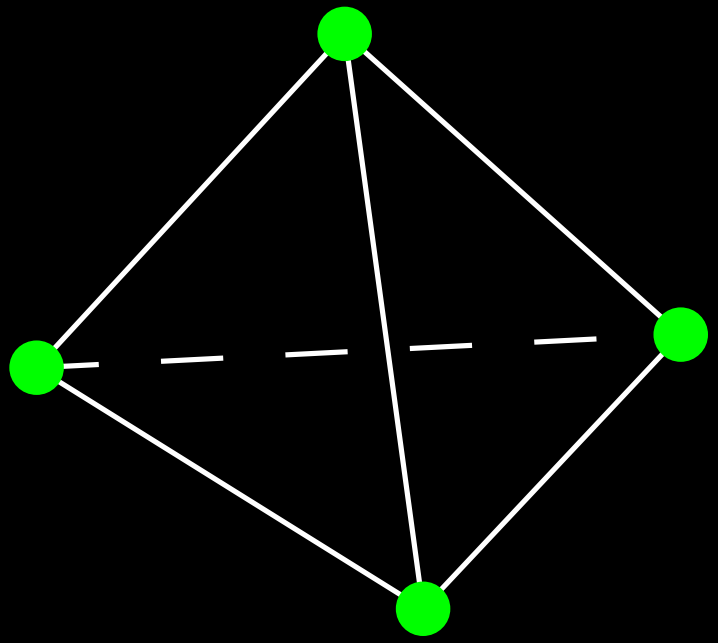
Vertices 頂点

- OpenGL ES draws triangles, lines, and points
- Object space vertices mapped into window space
- Rasterize the shapes to get pixels

描画できるのは
三角形・線分・点

オブジェクト座標系から
ウィンドウ座標系に
写像される

写像された「形状」データは，最終的に，
「ピクセル」データに変換される



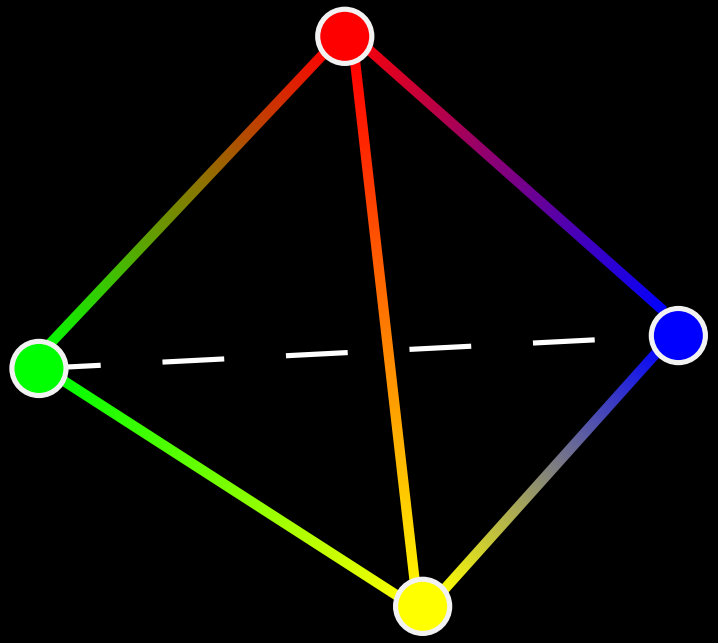
Colors

- Each vertex can have a color associated
- RGBA
  - Alpha usually means opacity
- Lines and triangles interpolate colors

各頂点は色をもつことが可能

A は透明度

線分や三角形は
色を「内挿」する



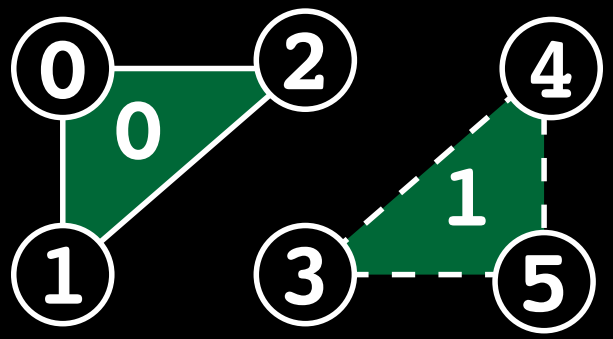
Drawing Modes 描画モード

- Vertices are passed to OpenGL ES in arrays 頂点配列が OpenGL へ
- Drawing modes determine how vertices are interpreted to produce shapes 描画モードは頂点配列の読み取り方を指定する
- Vertex order matters 与える頂点の順序が大切

Drawing Modes

- Vertices are passed to OpenGL ES in arrays
- Drawing modes determine how vertices are interpreted to produce shapes
- Vertex order matters

`GL_TRIANGLES`

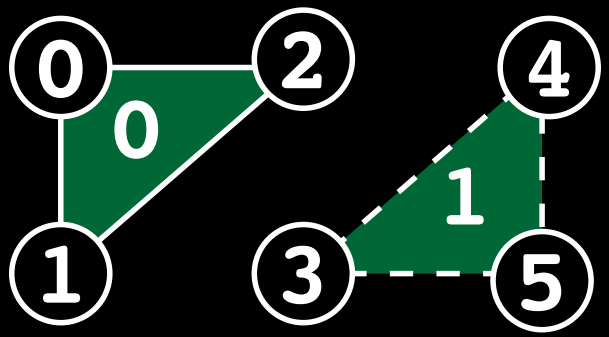


３頂点，３頂点，・・・

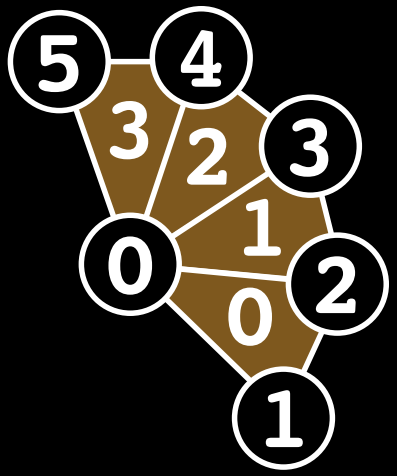
Drawing Modes

- Vertices are passed to OpenGL ES in arrays
- Drawing modes determine how vertices are interpreted to produce shapes
- Vertex order matters

GL_TRIANGLES



GL_TRIANGLE_FAN

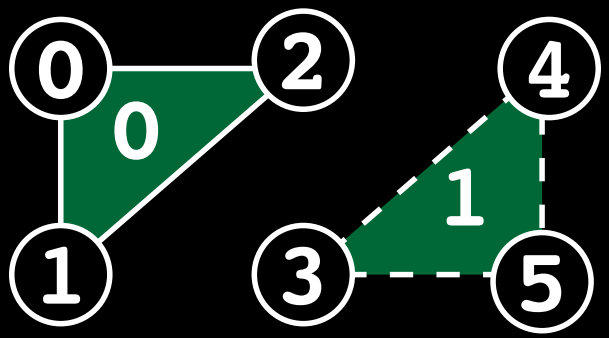


中心と扇型の各点

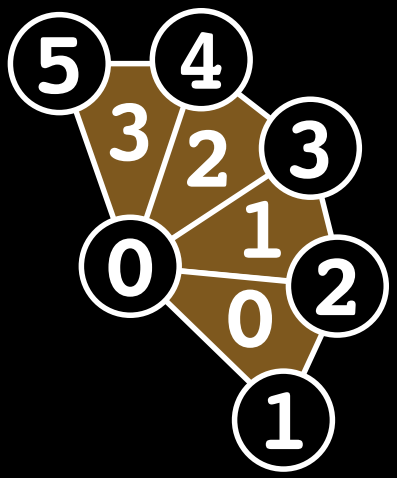
Drawing Modes

- Vertices are passed to OpenGL ES in arrays
- Drawing modes determine how vertices are interpreted to produce shapes
- Vertex order matters

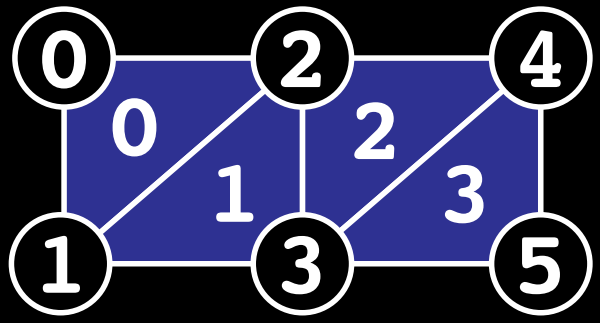
GL_TRIANGLES



GL_TRIANGLE_FAN



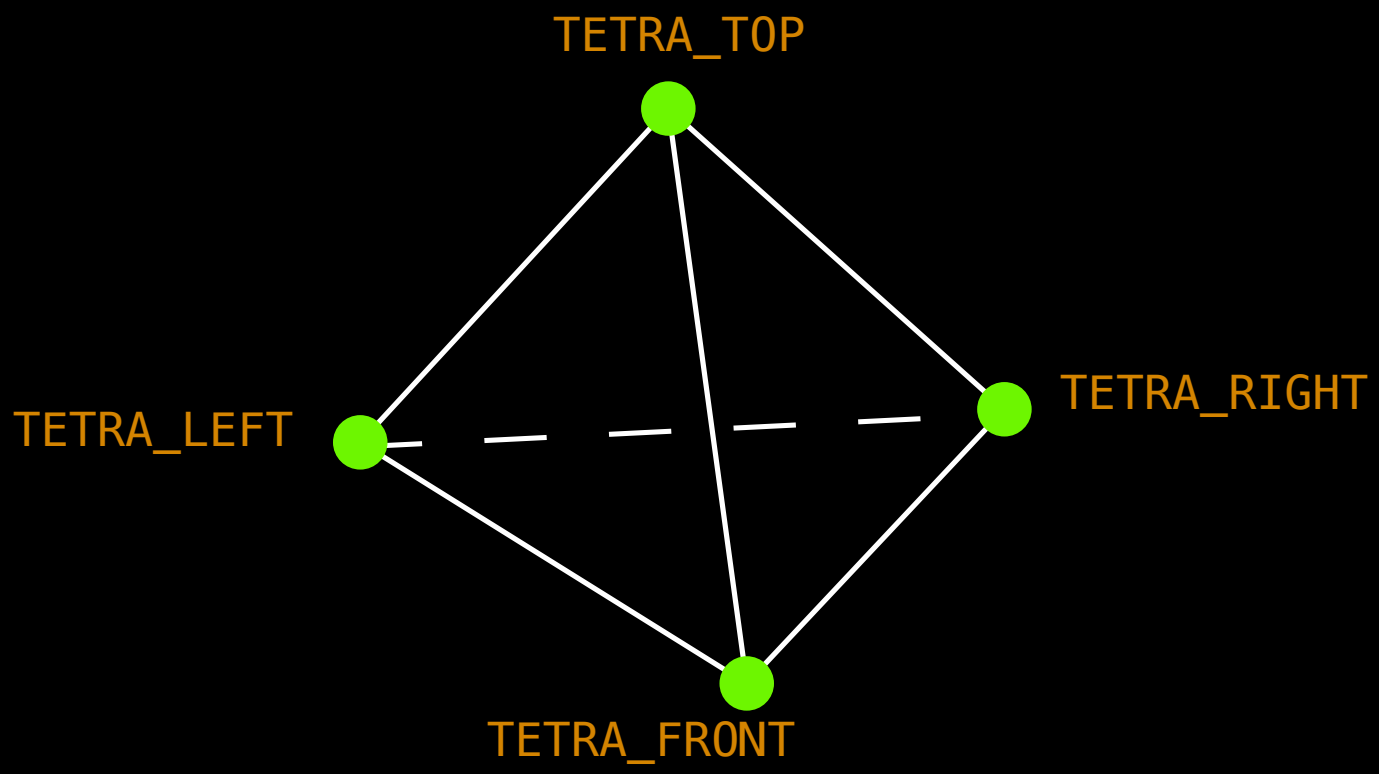
GL_TRIANGLE_STRIP



012, 123, 234, 345, ...

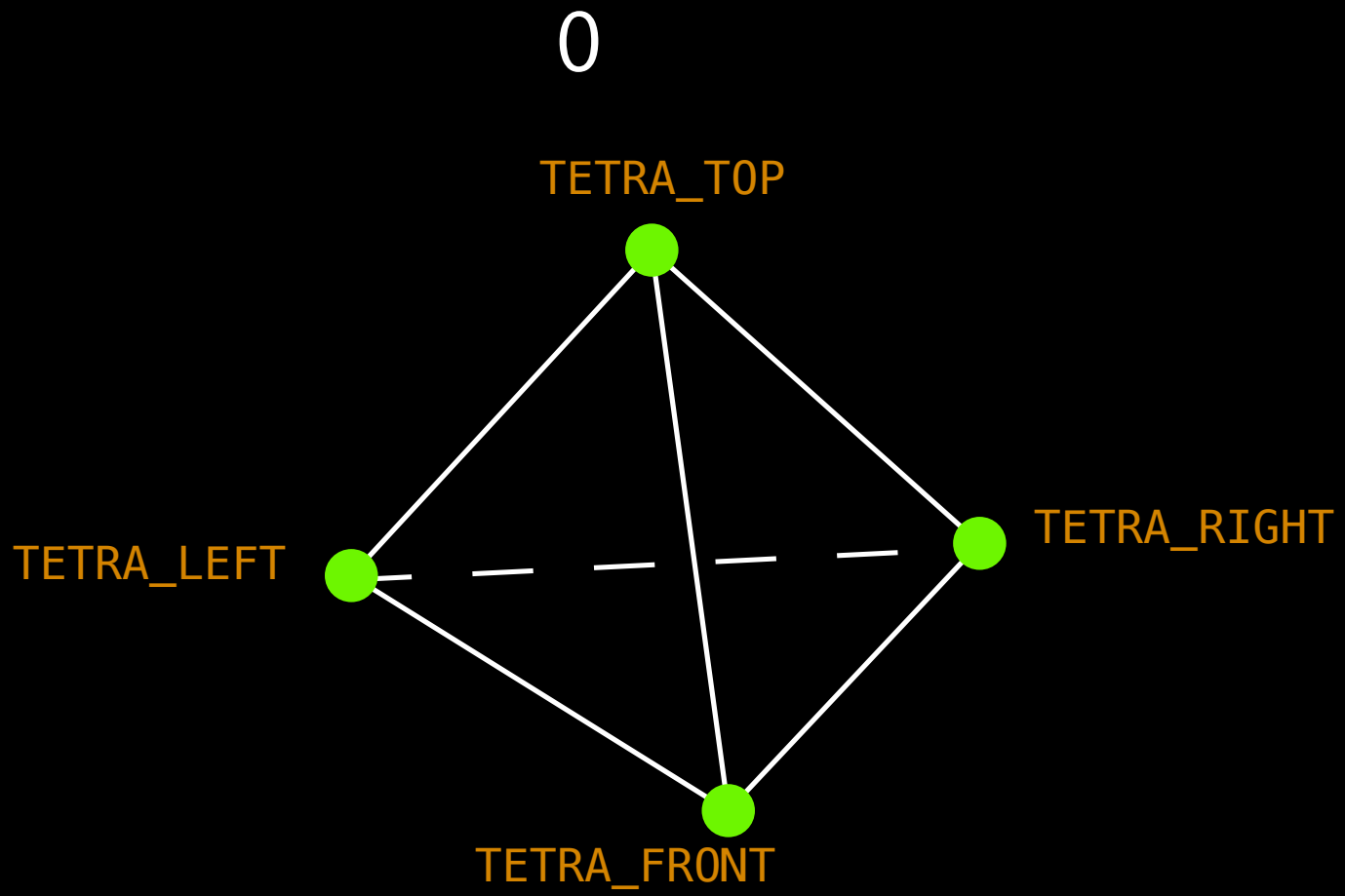
Drawing a Tetrahedron

- Use a triangle strip　**４面体を 012,123,234,…方式「一筆書」**



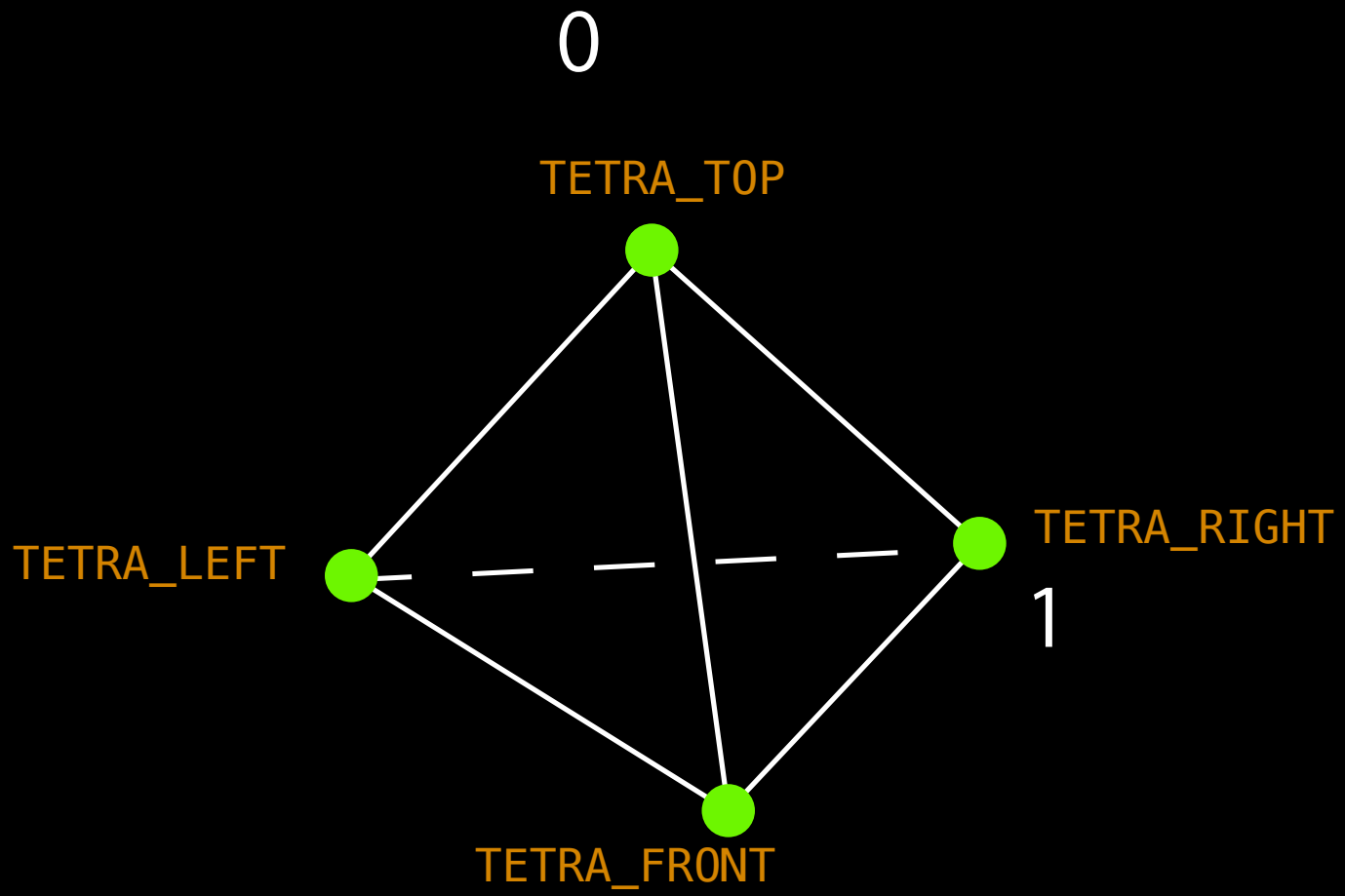
Drawing a Tetrahedron

- Use a triangle strip



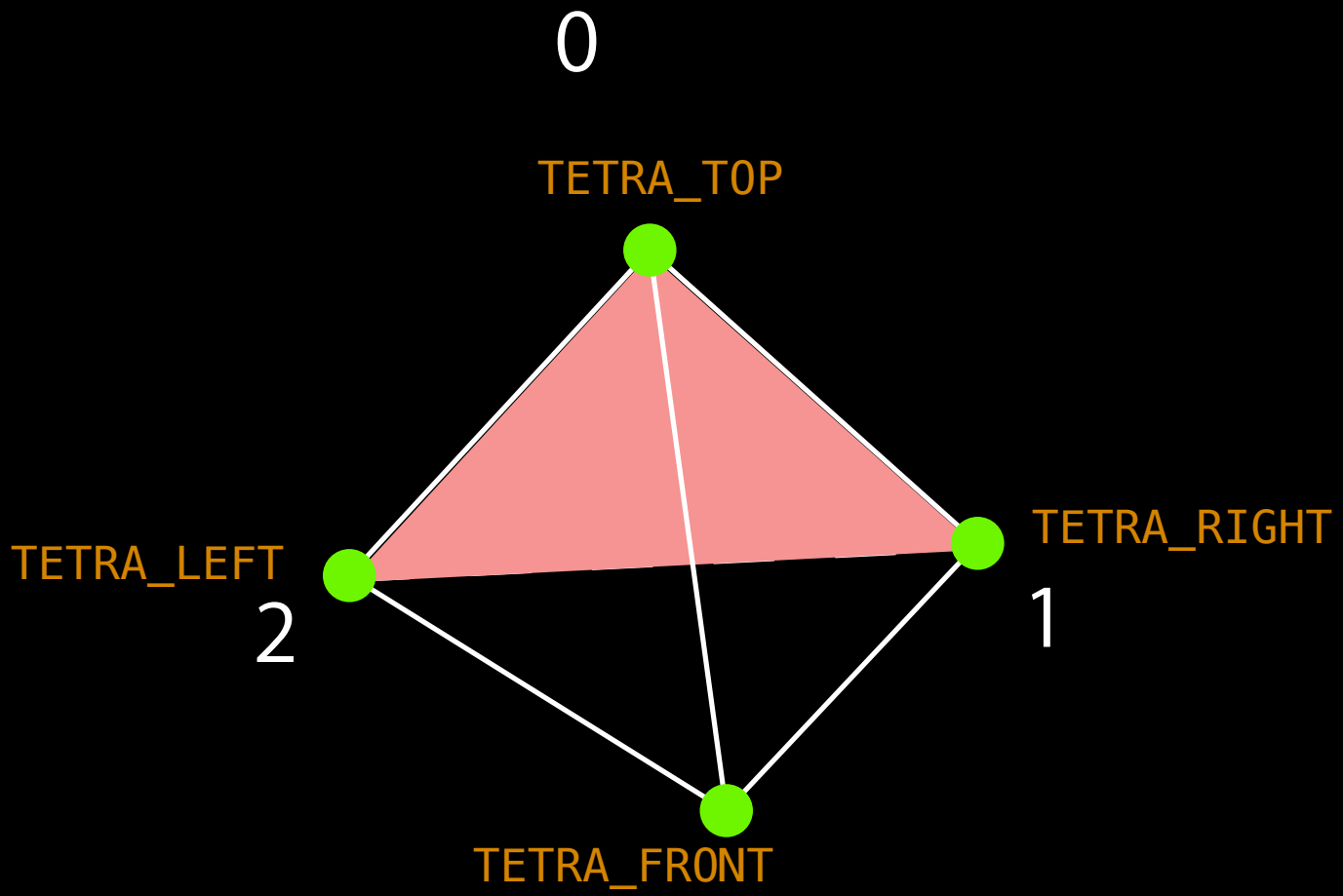
Drawing a Tetrahedron

- Use a triangle strip



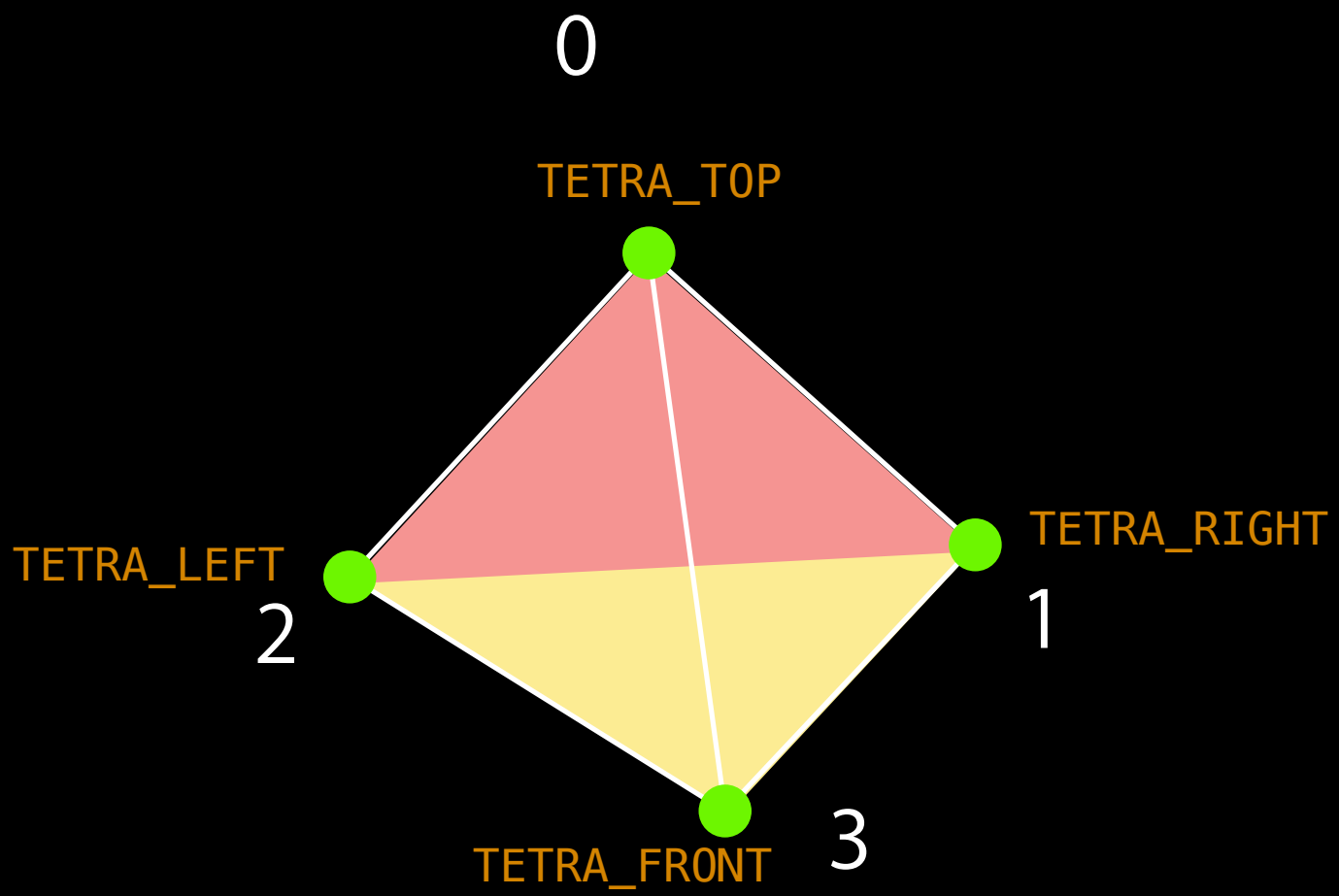
Drawing a Tetrahedron

- Use a triangle strip



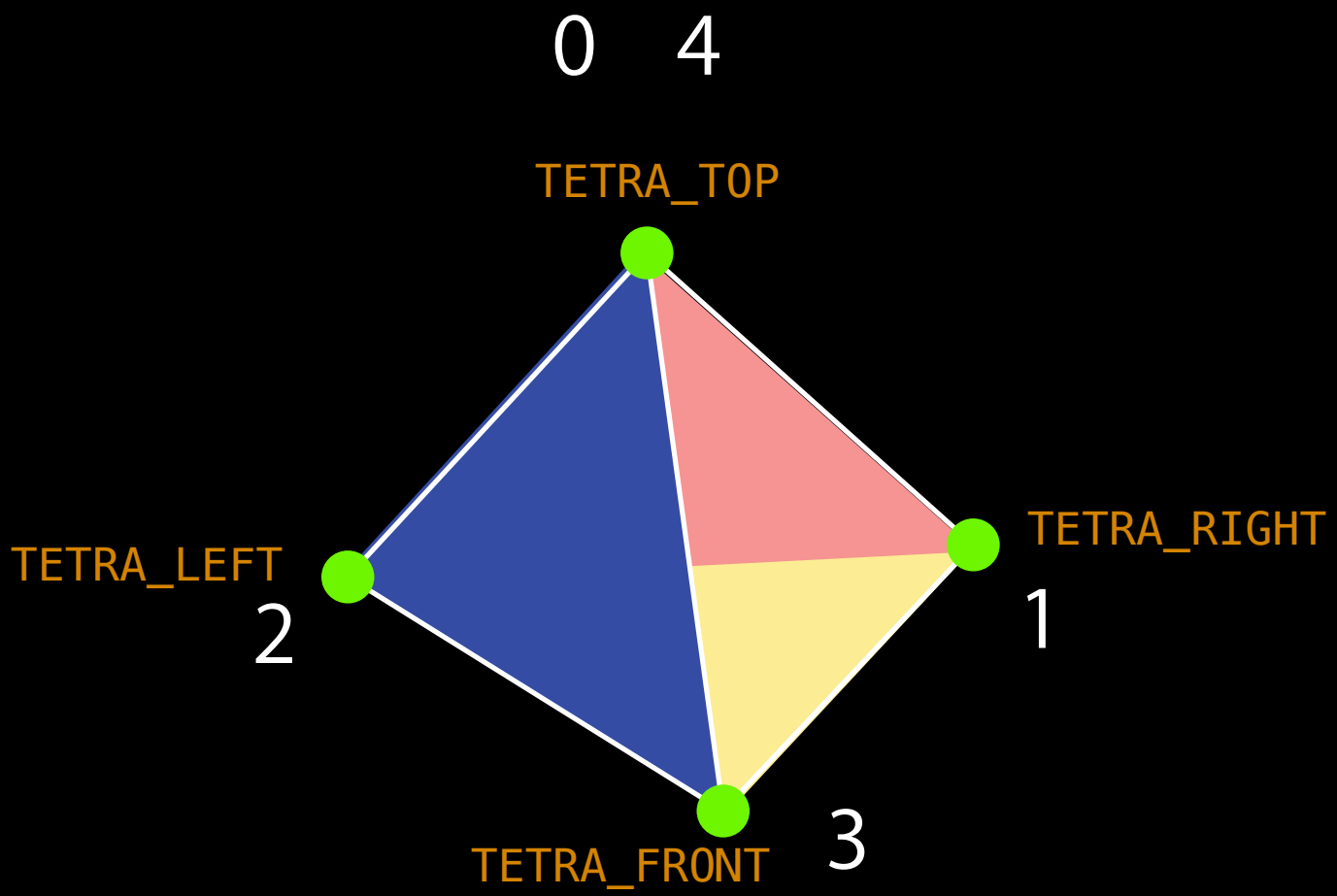
Drawing a Tetrahedron

- Use a triangle strip



Drawing a Tetrahedron

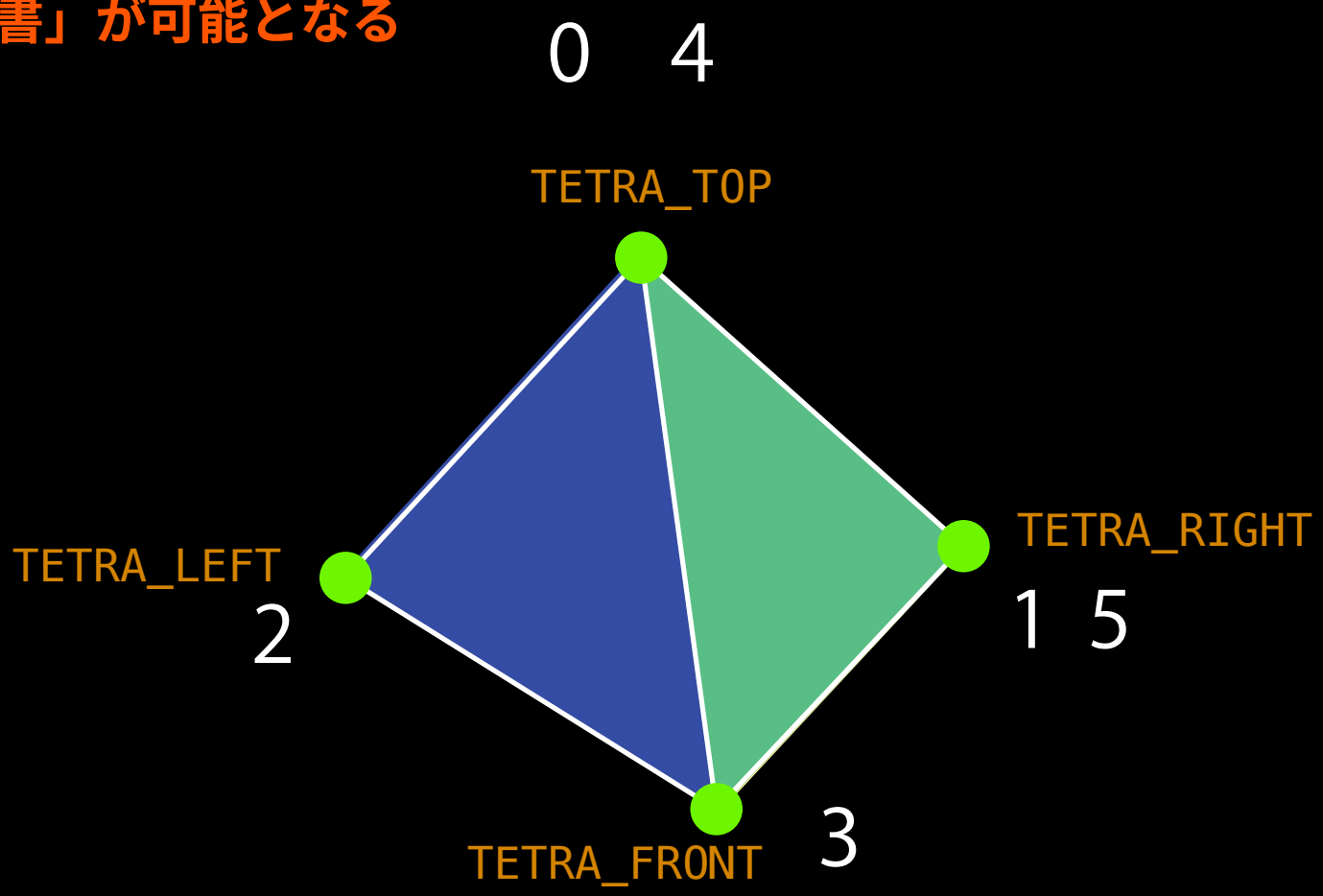
- Use a triangle strip



Drawing a Tetrahedron

- Use a triangle strip

同一座標に複数の頂点を置くことで
「一筆書」が可能となる



Demo

Geometry

Geometry Cheat Sheet

```
GLfloat vertexArray[] = {
    x1,y1,z1,
    x2,y2,z2,
    x3,y3,z3, ... };

GLubyte colorArray[] = {
    r1,g1,b1,a1,
    r2,g2,b2,a2,
    r3,g3,b3,a3, ... };

glVertexPointer(dimOfVertices, GL_FLOAT, arrayOffset, vertexArray);
glEnableClientState(GL_VERTEX_ARRAY);
glColorPointer(4, GL_UNSIGNED_BYTE, arrayOffset, colorArray);
glEnableClientState(GL_COLOR_ARRAY);

glDrawArrays(GL_TRIANGLE_STRIP, arrayOffset, numberOfVertices);
   // or GL_TRIANGLE_FAN, GL_TRIANGLES

glDisableClientState(GL_VERTEX_ARRAY);
glDisableClientState(GL_COLOR_ARRAY);
```

Using Textures

テクスチャ（画像）の貼り付け

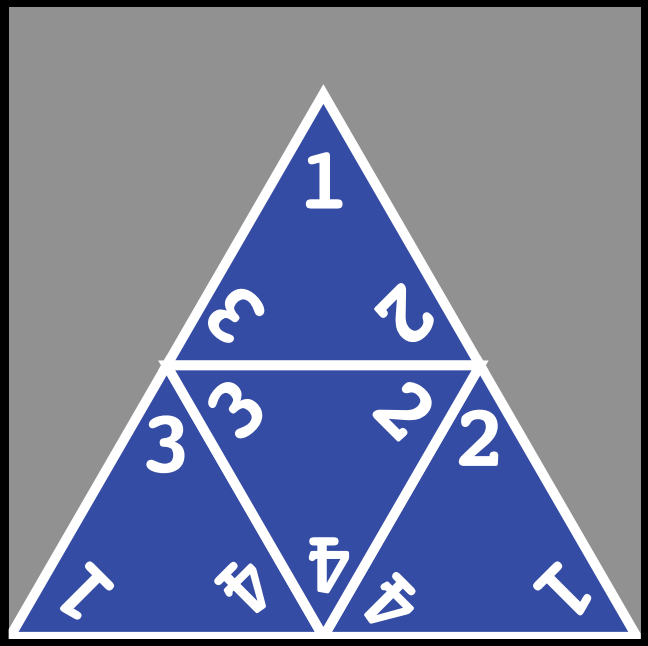
Texture Mapping

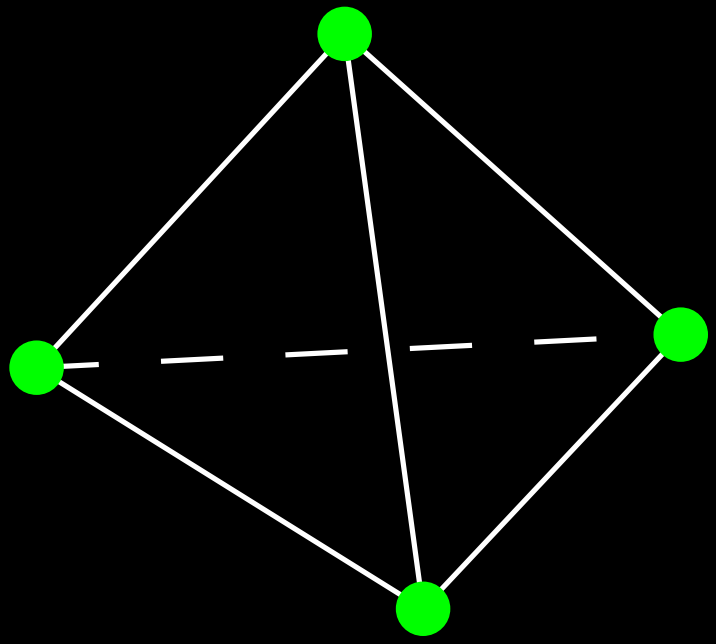
メモリ上の画像を
（三角形などの）面に
貼付ける

- Color pixels according to an image in memory
- Almost always 2D (3D textures are possible though) たいてい２D画像
- Vertices are given texture coordinates (u,v)

Texture Atlasing テクスチャの「地図」

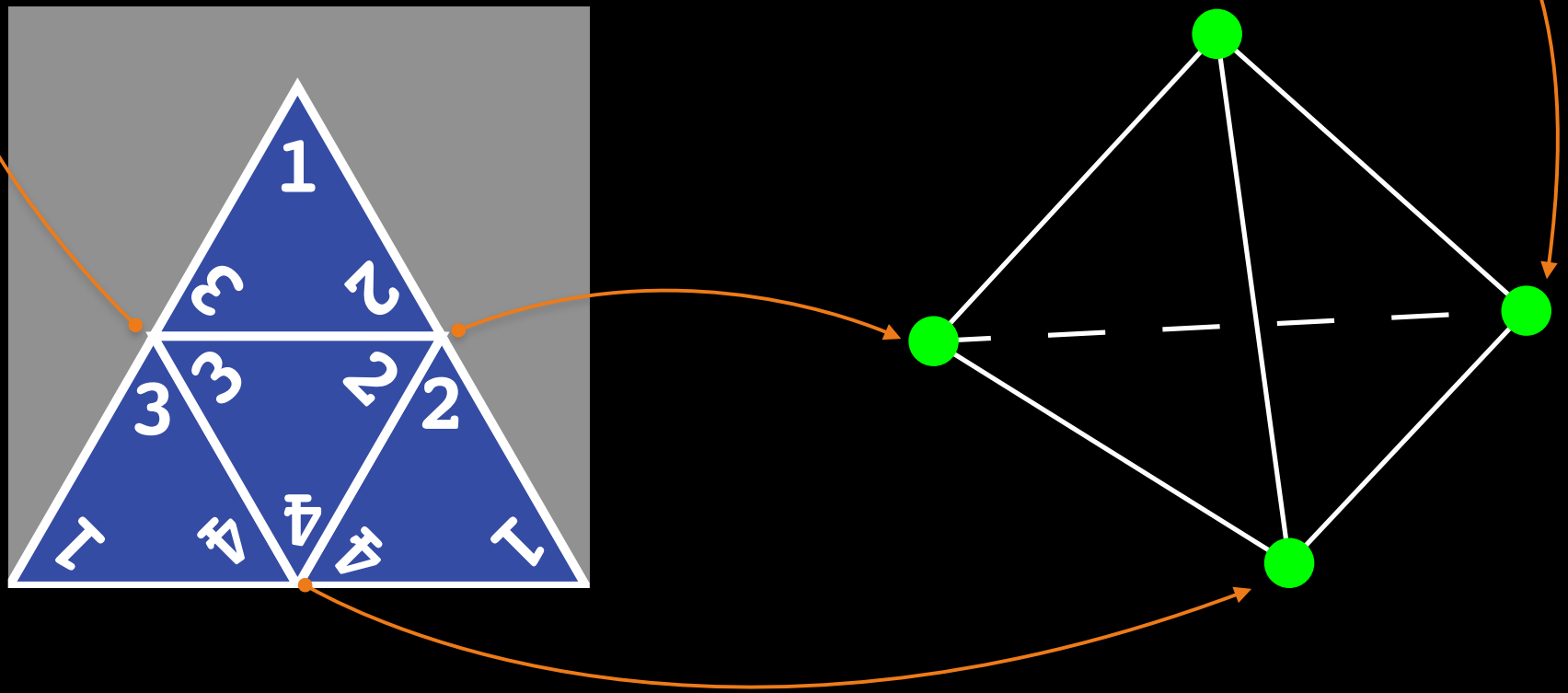
- Swapping textures often is inefficient テクスチャの張り替えは非効率
- Instead make one giant shared texture １枚の大画面から
  切り出して貼付けるとよい





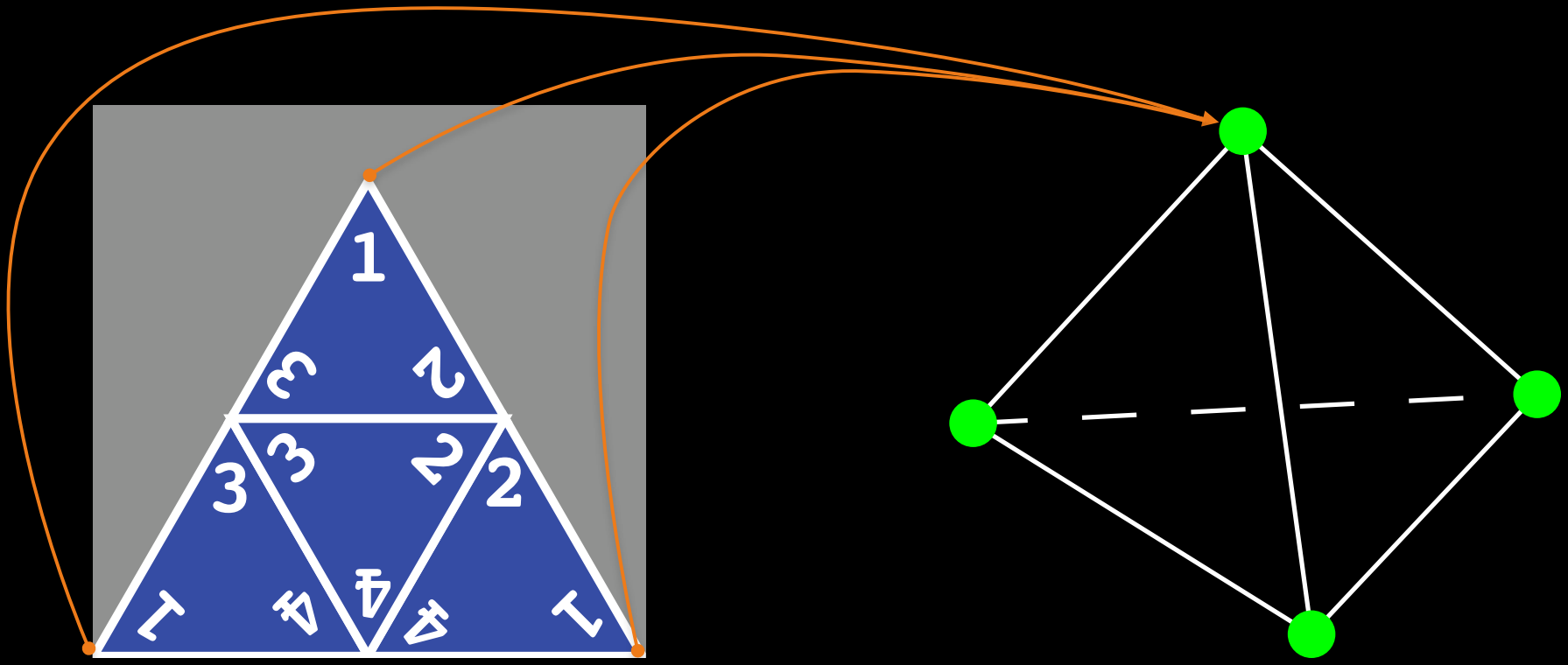
Texture Atlasing

- Swapping textures often is inefficient
- Instead make one giant shared texture



Texture Atlasing

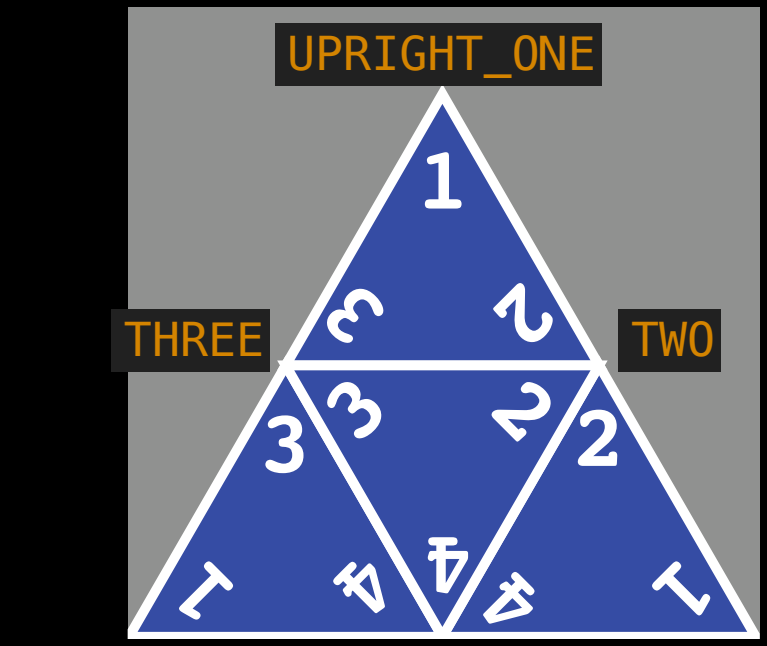
- Swapping textures often is inefficient
- Instead make one giant shared texture

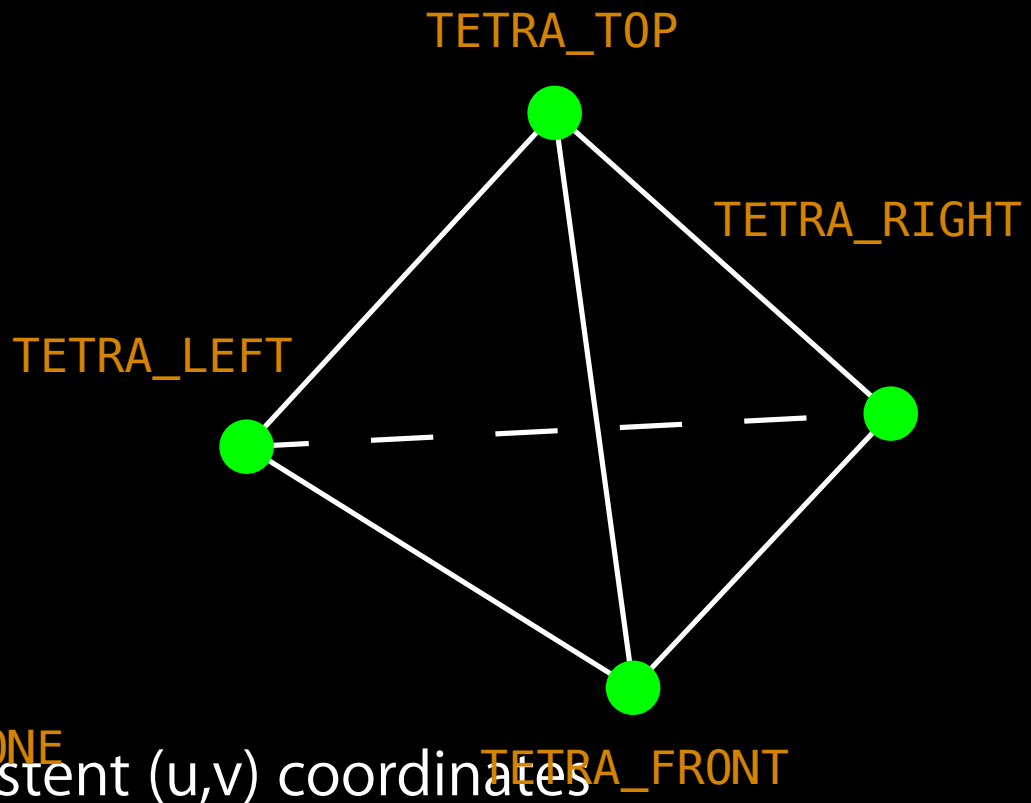


- Vertices may not have consistent (u,v) coordinates

Texture Atlasing

- Swapping textures often is inefficient
- Instead make one giant shared texture





- Vertices may not have consistent (u,v) coordinates

Demo

Textures

Texture Cheat Sheet

```
bash$ export PATH=${PATH}:/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/
bash$ texturetool -f PVR -e PVRTC image.png -o image.pvrtc
bash$ # image.png must be square with power of side length -- e.g. 64, 256, 1024

#import "PVRTexture.h"
// From Apple's PVRTextureLoader Example Project

NSString * path = [[NSBundle mainBundle] pathForResource: @"image" ofType:@"pvrtc"];
PVRTexture * texture = [[PVRTexture alloc] initWithContentsOfFile: path];

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAX_ANISOTROPY_EXT, 1.0f);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture.name);

GLFloat textureCoordArray[] = {
    u1,v1,
    u2,v2,
    u3,v3, ... };

glTexCoordPointer(2, GL_FLOAT, arrayOffset, textureCoordArray);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);
glDrawArrays(GL_TRIANGLE_STRIP, arrayOffset, numberOfVertices);
glDisableClientState(GL_TEXTURE_COORD_ARRAY);
```

Other Details

OpenGL ES 1.1 vs. ES 2.0

- This lecture described OpenGL ES1.1 ここでは 1.1 を扱った
- ES 2.0 is drastically different
  - Uses shader based approach
  - More flexible, harder to wrap your head around

2.0 は全く別モノ
シェーダー利用
柔軟だが難解

- ES 2.0 not available in iPhones before 3GS

2.0 は 3GS 以前では
使えない

Want to know more?

- Apple OpenGL Programming Guide
- OpenGL Redbook
- The Internets
- Stanford CS 148, 248

- Topics of interest
  - Framebuffers
  - Depth Testing
  - Backface Culling
  - Animation (not inherently part of OpenGL)
  - Transparency and Blending
  - Lighting and Shading

Questions?