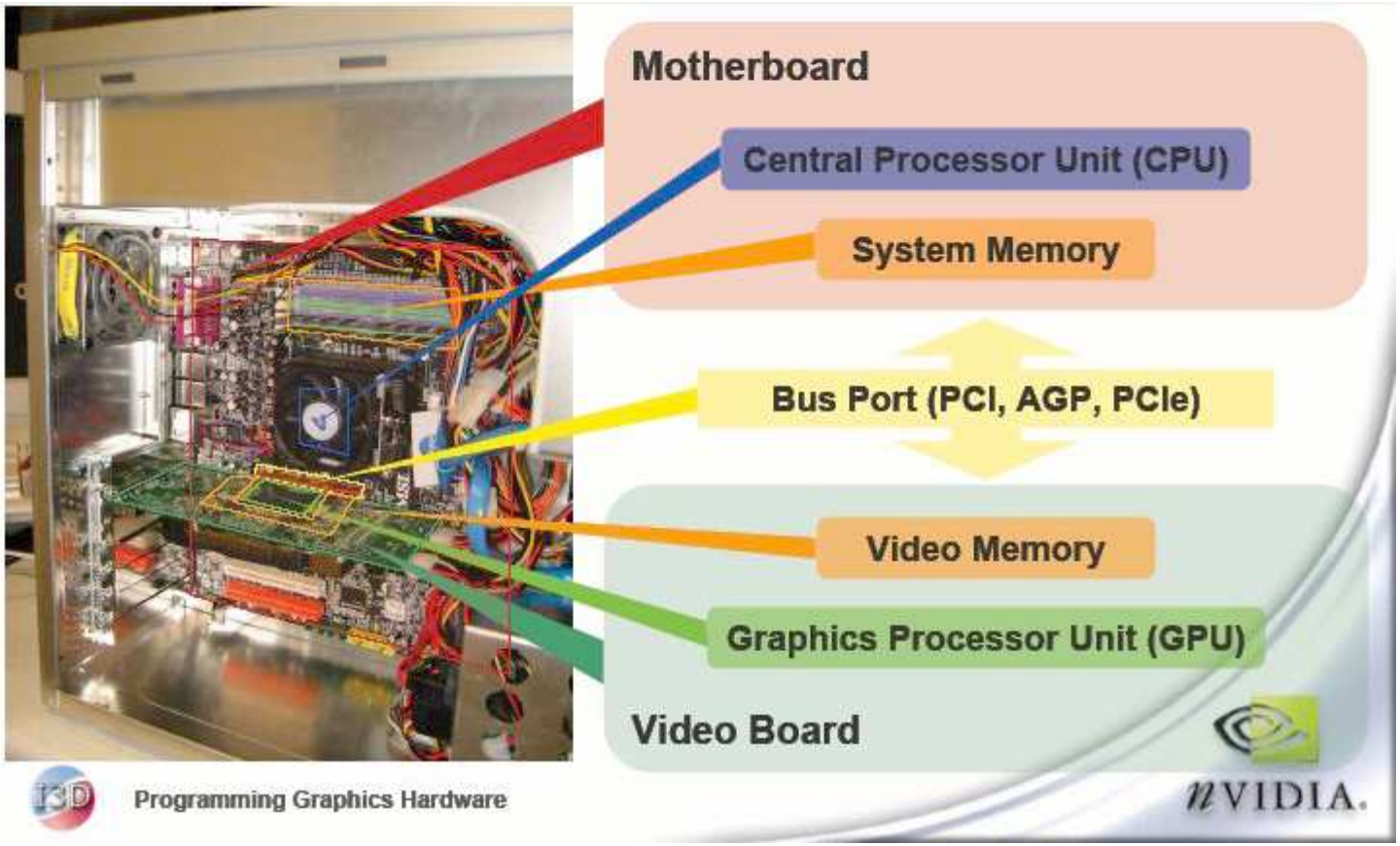
418341 สภาพแวดล้อมการทำงานคอมพิวเตอร์กราฟิกส์
การบรรยายครั้งที่ 2

ประมุข ขันเงิน
pramook@gmail.com

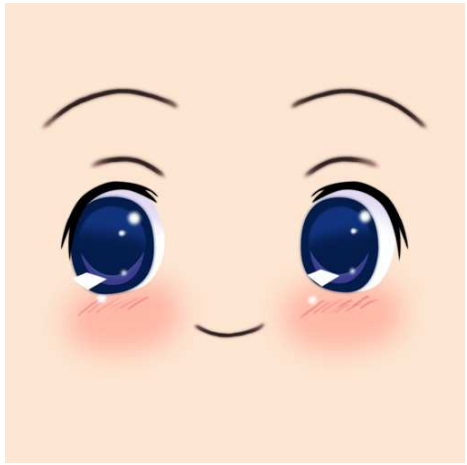
ในเครื่องคอมของคุณ...



รูปที่แล้วมีอะไรบ้าง?

- CPU, Memory, Bus
  - เรารู้ดีอยู่แล้วว่ามันทำอะไร
- GPU
  - ทำงานเกี่ยวกับกราฟิกส์
  - ข้อมูลเข้า
    - โมเดลรูปทรง (ตำแหน่งของจุด ความเชื่อมโยงกันของจุด และสีของจุด)
    - จิตรกรรมฝาผนัง
  - ข้อมูลออก
    - รูปบนหน้าจอ

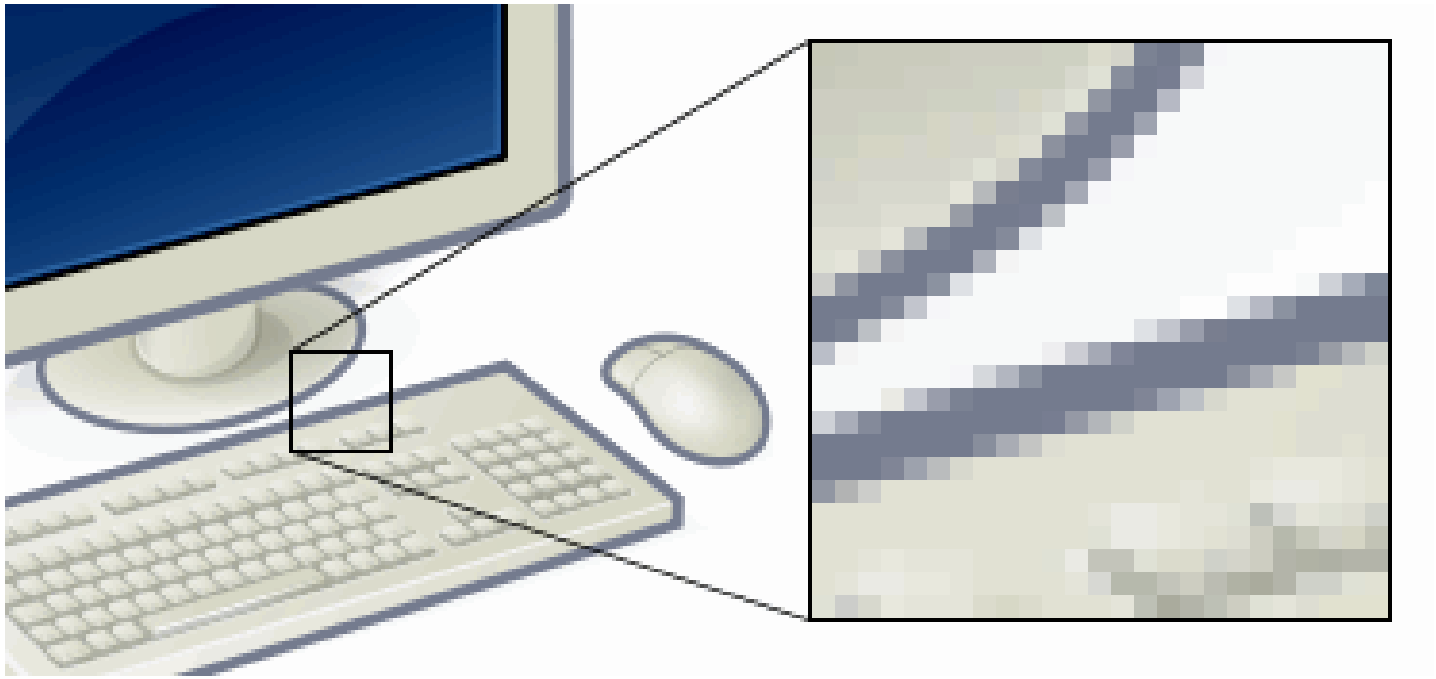
40.3765 246.3446 -13.3601
41.7488 226.0027 -5.0658
48.3294 235.3752 -7.3497
37.2949 230.1558 -9.6773
46.8526 239.2049 -10.7724
35.0925 232.2118 -10.9210
49.2234 231.9015 -5.4622
39.5274 227.7154 -6.8570
36.7923 240.2518 -18.0725
40.9546 241.5318 -16.3400
53.2942 227.1024 -17.4600
51.4157 231.8651 -20.9840
45.7685 234.6469 -25.0268
32.3952 239.7475 -5.4070
36.2495 235.5937 -5.3574
31.0568 236.1462 -9.5742
34.1015 253.4861 -8.2545
31.5805 251.6262 -9.3695
33.9048 256.8511 -4.1244



GPU

ภาพ

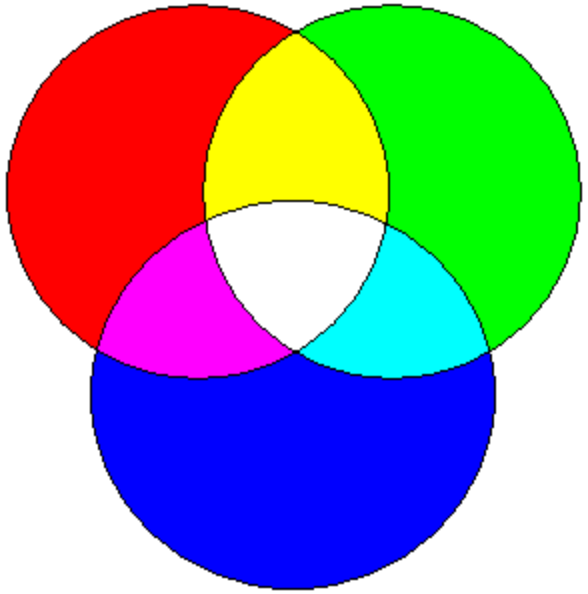
- ตารางสี่เหลี่ยมผืนผ้า แต่ละช่องมีสีหนึ่งสี
- แต่ละช่องเรียกว่า พิกเซล (pixel)



http://en.wikipedia.org/wiki/Pixels

สี

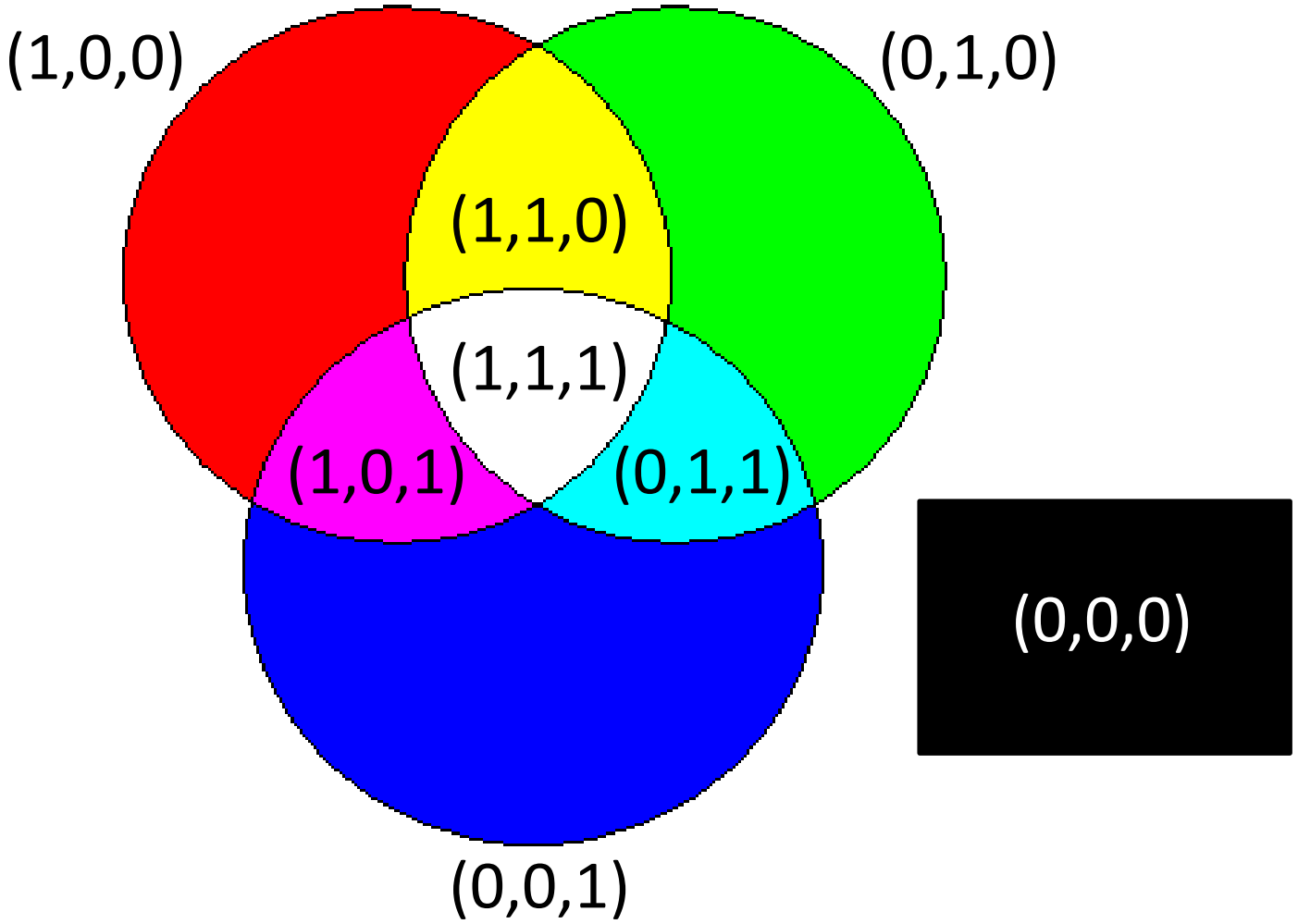
- สี = เวกเตอร์ (R, G, B) เลขแต่ละตัวมีค่าตั้งแต่ 0 ถึง 1
  - R บอกระดับความเข้มของแสงสีแดง
  - G บอกระดับความเข้มของแสงสีเขียว
  - B บอกระดับความเข้มของแสงสีน้ำเงิน



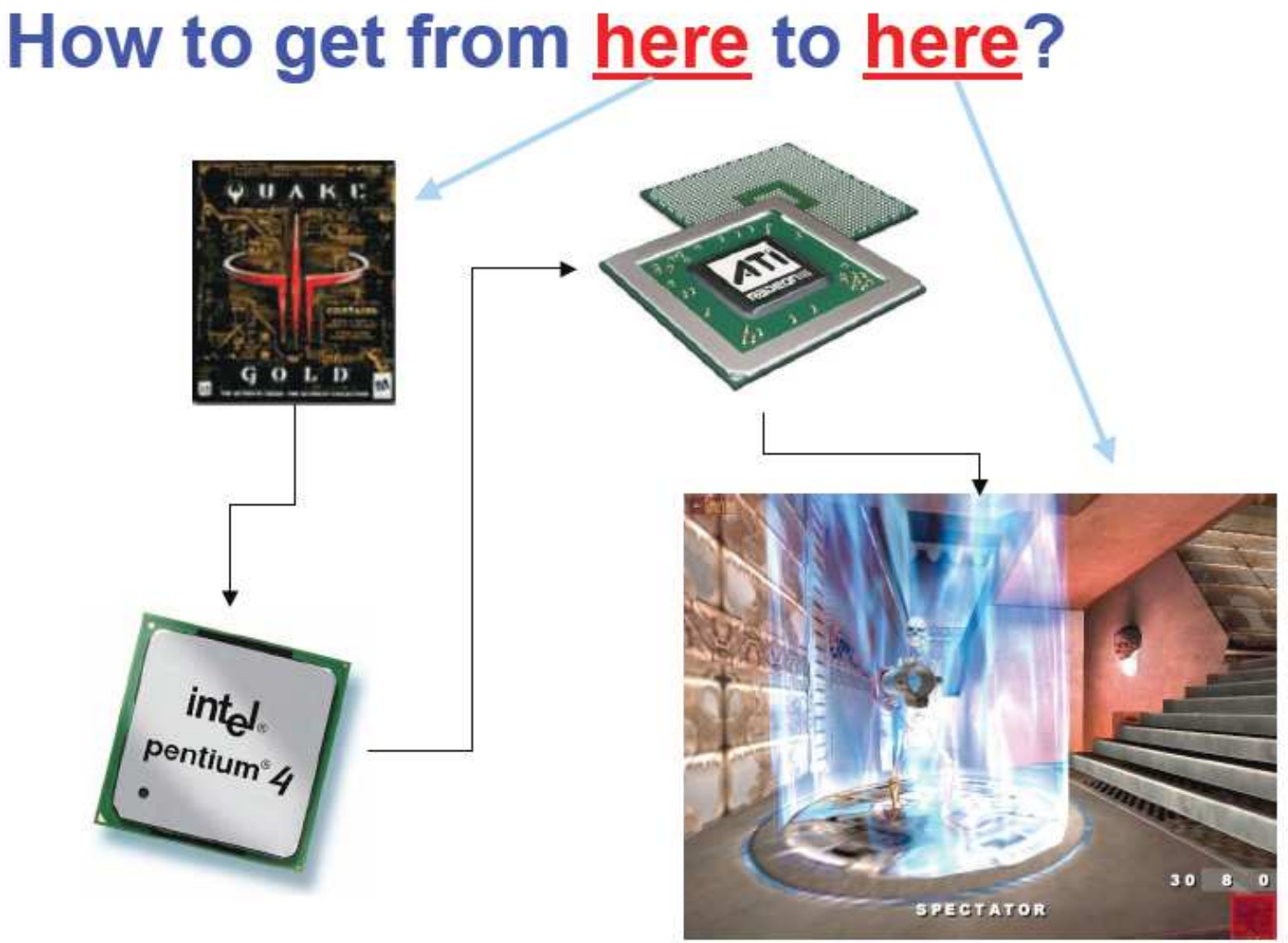
Trichromatic Theory of Vision

- สีที่มนุษย์มองเห็นแบ่งออกเป็นสามส่วน
  - แดง เขียว น้ำเงิน
  - ประสาทสัมผัสของมนุษย์ของแต่ละสีเป็นอิสระจากกัน
  - สีอื่นๆ เกิดจาก การนำสีทั้งสามนี้มาประกอบกัน
- หลักฐาน
  - เซลล์โคนในเรตินามีสามชนิด
  - แต่ละชนิดไวต่อ สีแดง สีเขียว สีน้ำเงิน ตามลำดับ

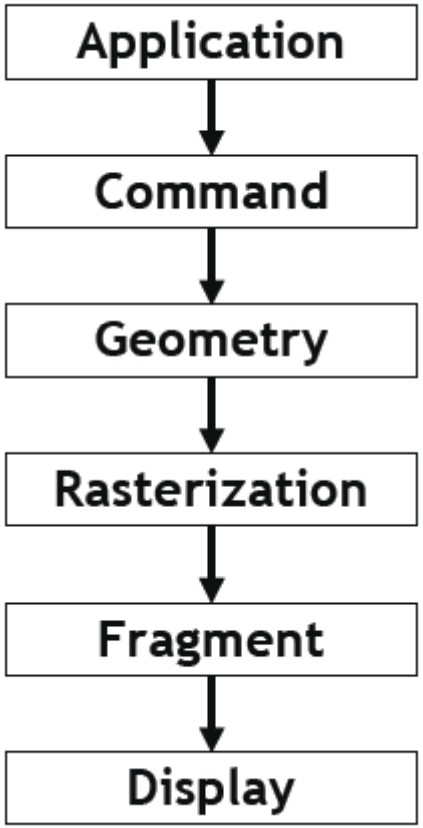
สีหลักๆ



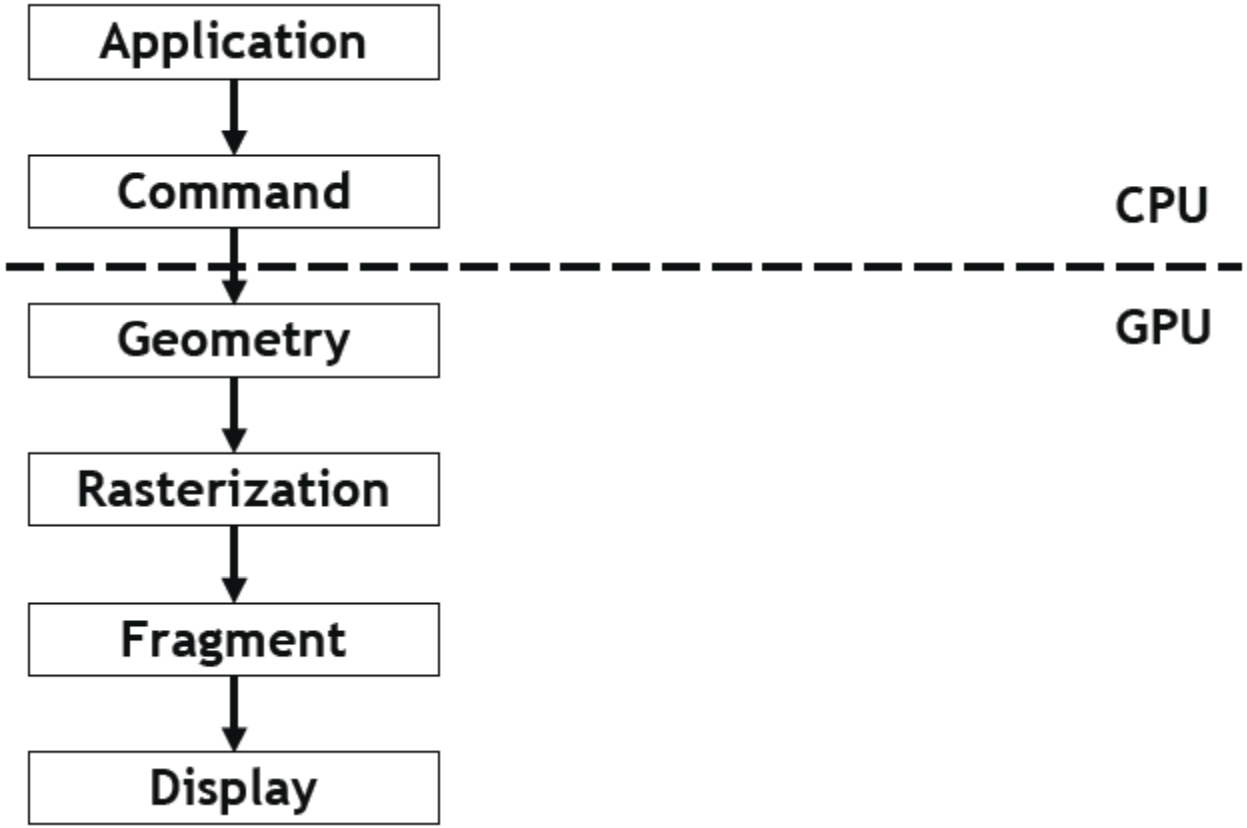
Graphics Pipeline



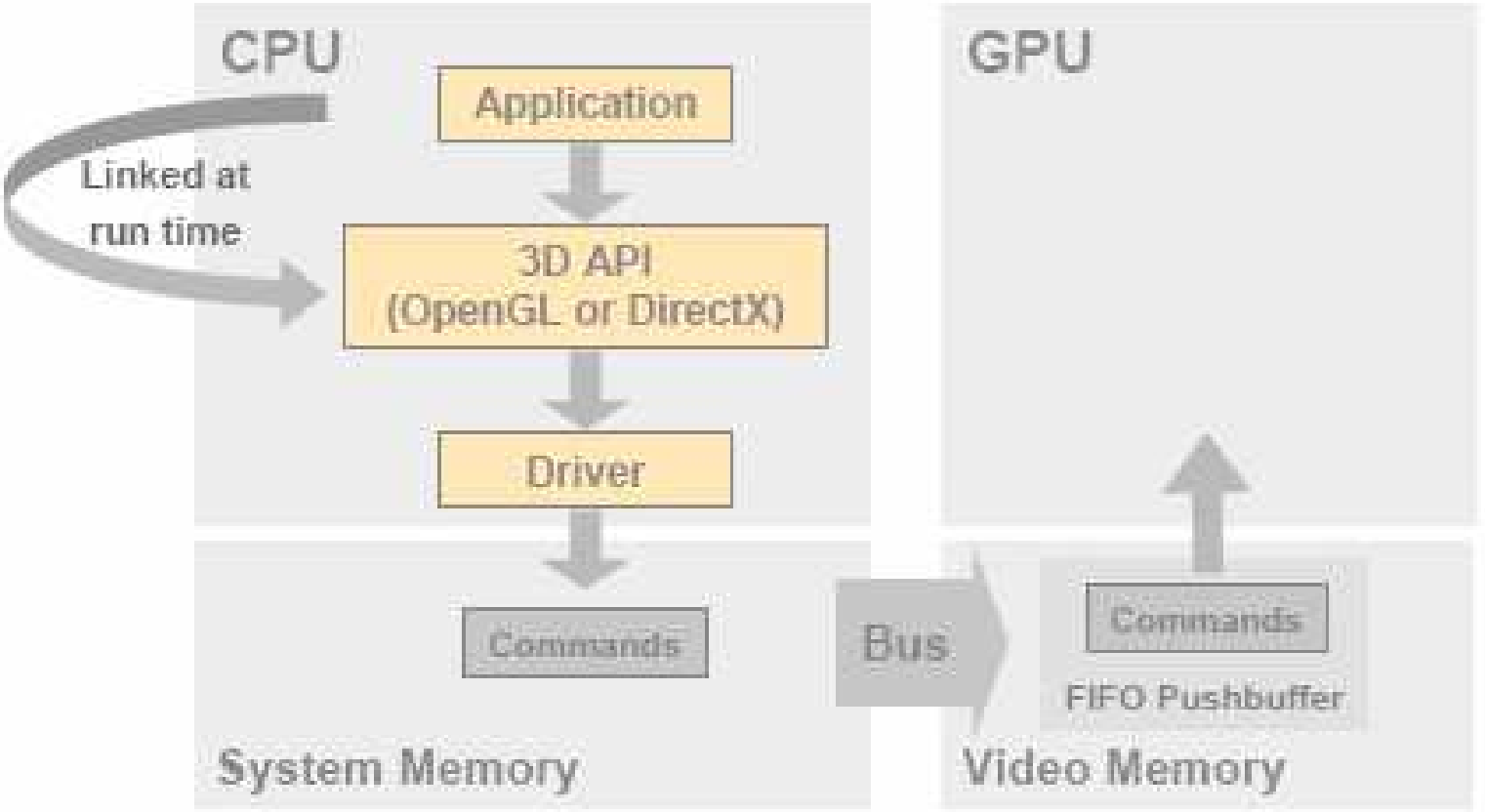
Graphics Pipeline (ต่อ)



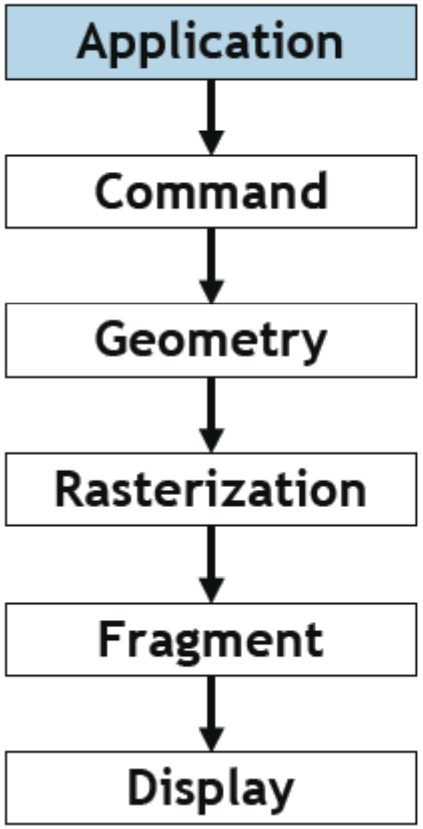
Graphics Pipeline (ต่อ)



Graphics Pipeline (ต่อ)

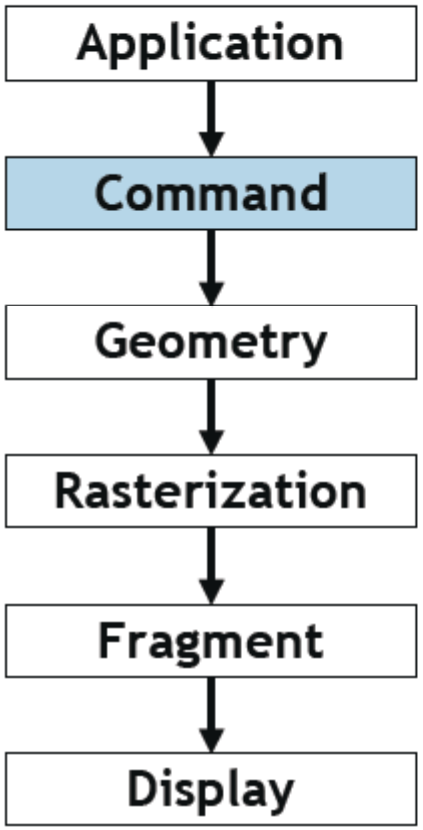


Graphics Pipeline (ต่อ)



- Quake 3:
- define game behavior
- networking
- user input events
- sound processing
- game AI
- game physics

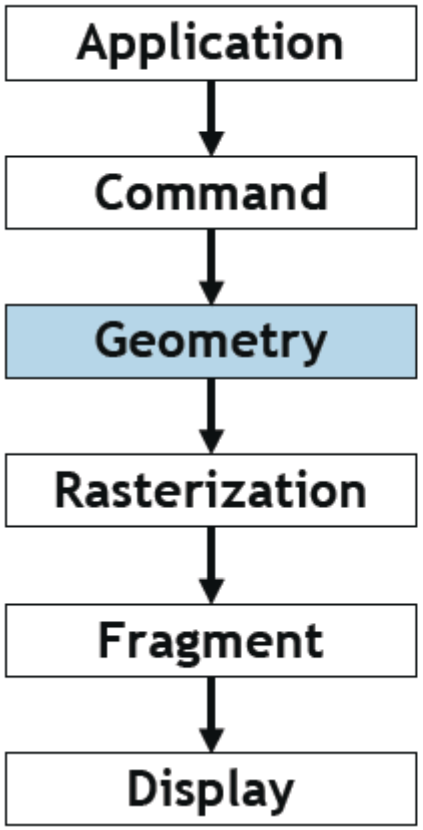
Graphics Pipeline (ต่อ)



- Quake 3:
- send OpenGL commands

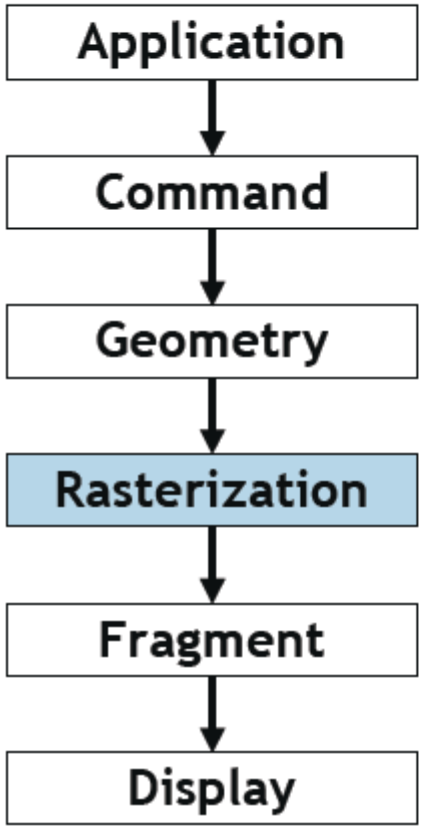
- OpenGL driver:
- process GL command stream
- talk to GPU

Graphics Pipeline (ต่อ)



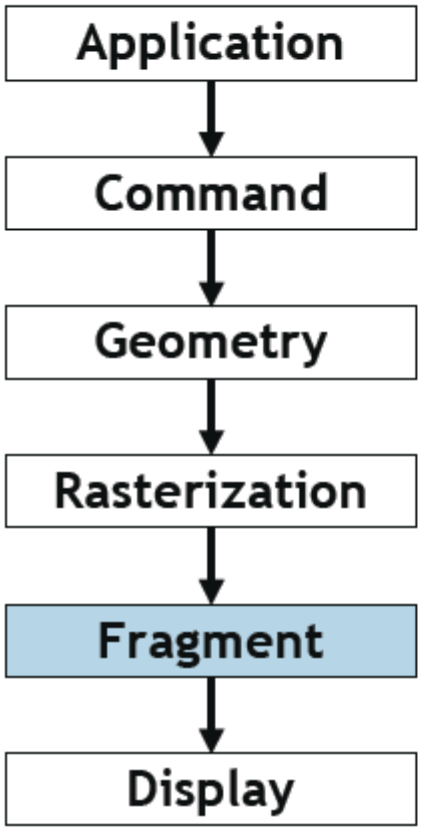
- GPU:
  - vertex transformations
  - vertex lighting
  - clipping
- primitive assembly

Graphics Pipeline (ต่อ)



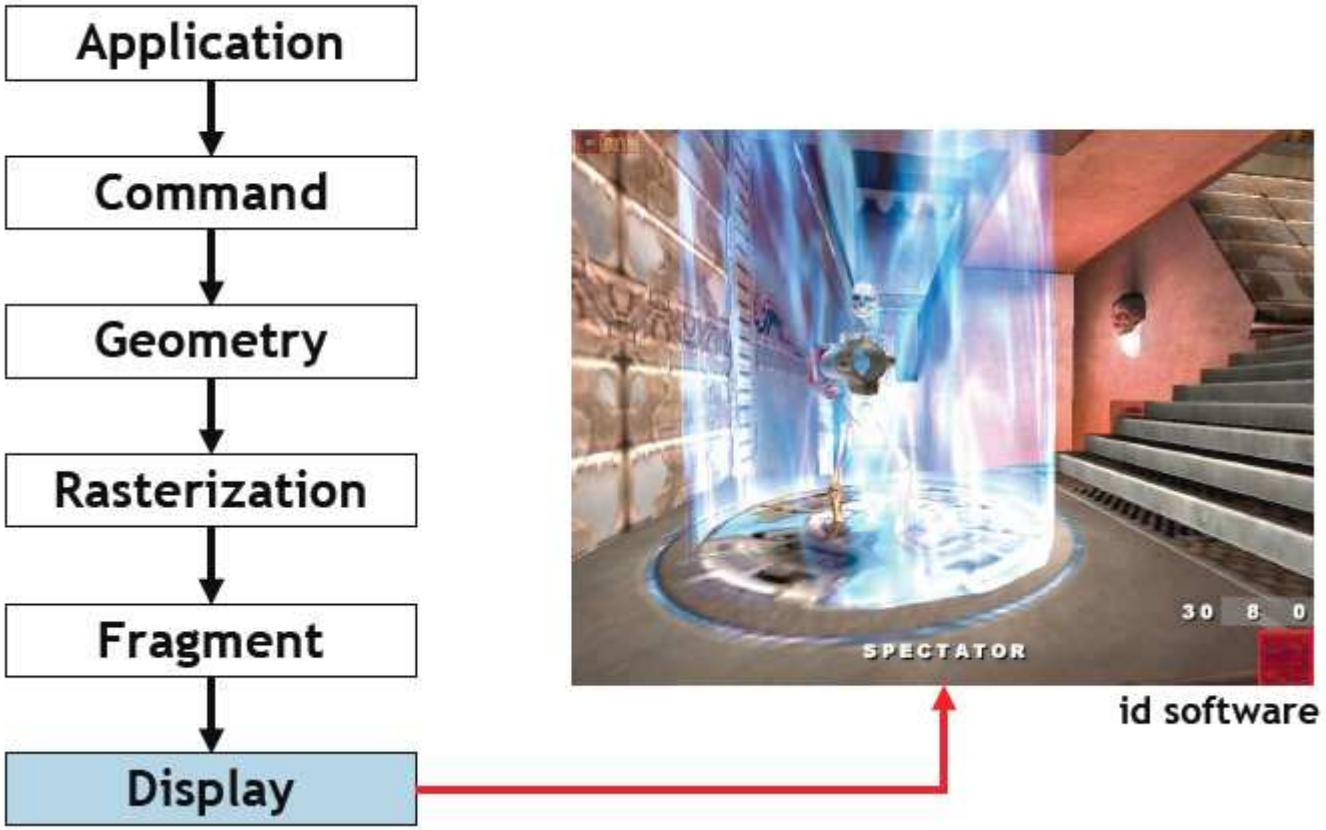
- GPU:
- convert triangles to fragments
- tex coordinate interpolation
- color interpolation

Graphics Pipeline (ต่อ)



- GPU:
- texturing
- depth test
- color blending

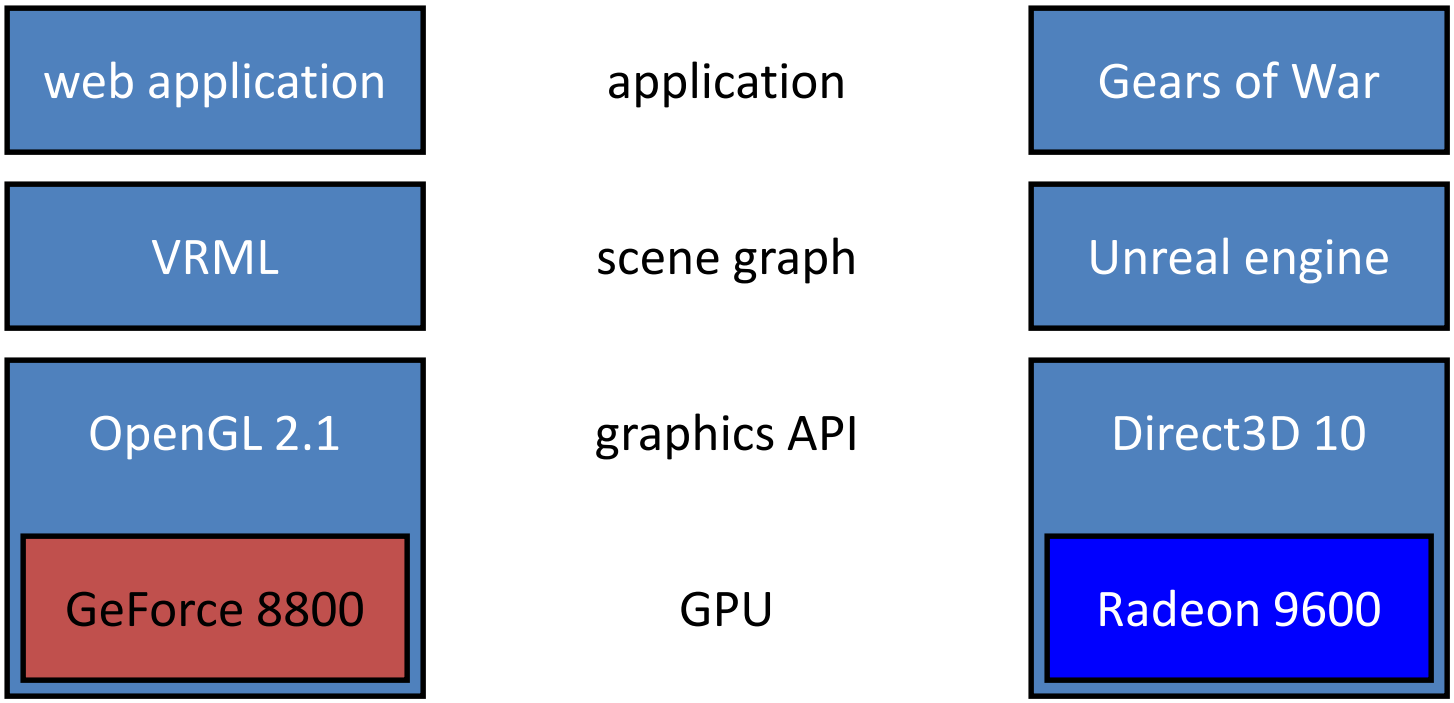
Graphics Pipeline (ต่อ)



OpenGL

- Application Programming Interface (API) สำหรับควบคุม GPU
- ผู้ใช้ OpenGL ระบุรูปทรงและรูปร่างพื้นฐาน (จุด เส้น และรูปหลายเหลี่ยม) ผ่านทาง OpenGL
- OpenGL จะทำหน้าที่สร้างภาพไว้บน framebuffer ให้
- ใช้สร้างโปรแกรมที่มีการตอบสนองต่อผู้ใช้แบบทันทีทันควัน (interactive) และโปรแกรมที่มีภาพเคลื่อนไหว
- ทำหน้าที่เดียวกับ Direct3D และเป็นคู่แข่งทางการค้ากันอยู่

กายวิภาคของโปรแกรมทางคอมพิวเตอร์กราฟฟิกส์



คำศัพท์

- Bitplane
  - เนื้อที่ในหน่วยความจำที่เก็บข้อมูล 1 บิทของทุกพิกเซลที่อยู่บนจอภาพ
- Framebuffer
  - Bitplane หลายๆ bitplane ที่เก็บข้อมูลทั้งหมดที่ใช้ควบคุมหน้าจอ
- Buffer
  - Bitplane กลุ่มหนึ่งที่ใช้เก็บข้อมูลบางอย่าง
- Application Programming Interface (API)
  - ฟังก์ชันและ object อื่นๆ ในภาษาระดับสูงที่ให้โปรแกรมประยุกต์ใช้สำหรับติดต่อกับระบบฮาร์ดแวร์หรือซอฟต์แวร์ต่างๆ

สิ่งที่ OpenGL ไม่ทำ

- จัดการการติดต่อกับผู้ใช้
- จัดการวินโดวส์
- วาดและจัดการรูปทรงที่ซับซ้อน เช่น รถถัง ต้นไม้ ฯลฯ
  - ถึงแม้ว่าคุณจะสามารถใช้รูปทรงง่ายๆ ของ OpenGL สร้างมันได้ก็ตาม
  - ส่วนใหญ่คุณต้องเขียน library ขึ้นมาจัดการกับพวกนี้เอง
- จัดการ framebuffer
  - เป็นความรับผิดชอบของคุณที่ต้องเตรียม framebuffer ให้ OpenGL

GLUT

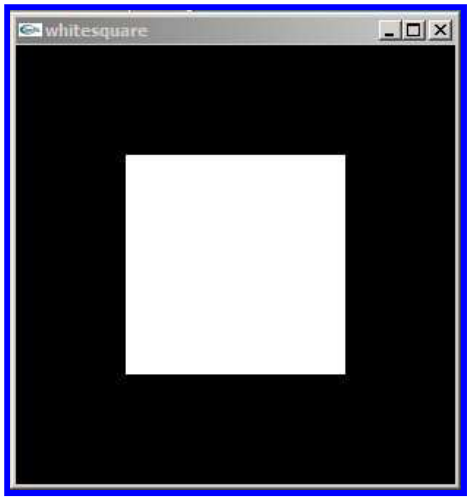
- OpenGL Utility Toolkit
- ใช้สำหรับจัดการการติดต่อกับผู้ใช้และจัดการวินโดวส์
  - ทำสิ่งที่ OpenGL ไม่ทำ
- เอาไปใช้เขียนโปรแกรมประยุกต์จริงๆ คงยาก
  - ไม่มี GUI Widget ให้ใช้เลย
  - ต้องรับข้อมูลจากผู้ใช้ตามที่ GLUT กำหนด
- แต่ทำให้การเรียนรู้ OpenGL ง่ายขึ้นมาก

ตัวอย่าง

```
// Draw a white square against a black background


#include <windows.h>
#include <stdio.h>
#include "glut.h"

void draw() {
    glClearColor(0.0, 0.0, 0.0, 0.0)
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f(-0.5, -0.5, 0.0);
        glVertex3f( 0.5, -0.5, 0.0);
        glVertex3f( 0.5,  0.5, 0.0);
        glVertex3f(-0.5,  0.5, 0.0);
    glEnd();
    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("whitesquare");
    glutDisplayFunc(draw);
    glutMainLoop();
}
```

OpenGL

GLUT



เฉพาะส่วนของ OpenGL

```
glClearColor(0.0, 0.0, 0.0, 0.0)
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_POLYGON);
  glVertex3f(-0.5, -0.5, 0.0);
  glVertex3f( 0.5, -0.5, 0.0);
  glVertex3f( 0.5,  0.5, 0.0);
  glVertex3f(-0.5,  0.5, 0.0);
glEnd();
glFlush();
```

ทีละคำสั่ง

- glClearColor(0.0, 0.0, 0.0, 0.0)
  - เปลี่ยนสีที่จะใช้ล้างหน้าจอเป็นสีดำ
- glClear(GL_COLOR_BUFFER_BIT)
  - ล้าง bitplane ที่เก็บสีด้วยสีที่กำหนดใน glClearColor
- glColor3f(1.0, 1.0, 1.0)
  - เปลี่ยนสีเป็นสีขาว
  - จุดที่วาดต่อจากนี้ไปจะเป็นสีขาว

ทีละคำสั่ง (ต่อ)

- glBegin(GL_POLYGON)
  - บอกว่าต่อไปเราจะวาดรูปหลายเหลี่ยม
- glVertex3f(x, y, z)
  - กำหนดจุด
- glEnd()
  - บอกว่าสิ่งที่เริ่มไปตั้งแต่ glBegin ที่แล้วได้เสร็จสิ้นแล้ว
  - ในที่นี้คือบอกว่ากำหนดรูปหลายเหลี่ยมเสร็จแล้ว
- glFlush()
  - ทำให้คำสั่ง OpenGL ที่เคยสั่งมาถูกนำไปปฏิบัติงาน แทนที่จะถูกเก็บไว้ในหน่วยความจำเพื่อรอคำสั่งอื่น

คำสั่ง OpenGL

- ขึ้นต้นด้วย gl
- ตามด้วยชื่อคำสั่ง เช่น Vertex หรือ Color
- บางคำสั่งอาจมี*จำนวนและชนิด*ของ argument
  - 3f บอกว่าต้องการ argument เป็น float 3 ตัว
    - glVertex3f(1.0f, 3.0f, 4.0f);
  - 2i บอกว่าต้องการ argument เป็น int 2 ตัว
    - glVertex2i(-1, 5);
  - 3fv บอกว่าต้องการ argument เป็น pointer ไปยัง float 3 ตัว
    - float colorArray[] = {1.0f, 0.0f, 0.0f}
    - glColor3fv(colorArray);

ชนิดของ argument ในชื่อคำสั่ง

Suffix	Data Type	Typical Corresponding C-Language Type	OpenGL Type Definition
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	long	GLint, GLsizei
f	32-bit floating-point	float	GLfloat, GLclampf
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum, GLbitfield

OpenGL เป็น State Machine

- OpenGL จะจำค่าต่างๆ ที่ผู้ใช้กำหนดได้เอาไว้
- เมื่อผู้ใช้กำหนดค่า ค่านั้นจะถูกใช้ต่อไปเรื่อยๆ จนกว่าจะเปลี่ยน
- ค่าที่จำไว้ เช่น
  - สีที่ใช้ล้างหน้าจอ
  - สีของจุด
  - ทิศทางและตำแหน่งของกล้องถ่ายรูป
- ยกตัวอย่างเช่น เวลาเราเรียก glColor3f(1,1,1) แล้วสีของจุดที่กำหนดด้วย glVertex จะเป็นสีขาวไปจนกว่าจะเรียก glColor ใหม่อีกครั้ง

ซอฟต์แวร์ที่เราจะใช้

- Microsoft Visual C++ Express 2008
  - http://www.microsoft.com/express
- GLUT for Win32
  - http://www.xmission.com/~nate/glut.html

การใช้ OpenGL และ GLUT

- เวลาใช้ OpenGL ต้องมีการ include header ต่อไปนี้
  - #include <GL/gl.h>
  - #include <GL/glu.h>
- เวลาใช้ GLUT ต้องมีการ include header ต่อไปนี้
  - #include “glut.h”
- เนื่องจาก GLUT ทำการ include gl.h และ glu.h ให้เราอยู่แล้ว จึงไม่จำเป็นต้องพิมพ์สองบรรทัดบนถ้ามีบรรทัดที่สาม

โค้ดตัวอย่างเฉพาะส่วนของ GLUT

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("whitesquare");
    glutDisplayFunc(draw);
    glutMainLoop();
}
```

glutInit

- glutInit(int *argc, char **argv)
  - ทำการตั้งค่าเริ่มต้นหลายๆ ค่าของ GLUT
  - สิ่งที่ต้องส่งให้คือ pointer ไปยังจำนวน argument ของโปรแกรม และ argument อื่นๆ
  - ต้องเรียกเป็นคำสั่งแรกก่อนคำสั่งอื่นของ GLUT ทั้งหมด
  - ความจริงไม่มีอะไรมาก ปกติเราเขียน
    int main(int argc, char ** argv)
  - ก็แค่ให้เรียก glutInit(&argc, argv) เป็นคำสั่งแรกใน main ก็พอ

glutInitDisplay

- glutInitDisplay(unsigned int mode)
  - เลือกว่าสีของ pixel ในโปรแกรมของเราจะเป็นแบบใด
    - มีให้เลือกสองแบบคือ RGB กับ Indexed Color
    - เราจะไม่ใช้ Indexed Color เลย
  - เลือกว่าจะใช้ single buffer หรือ double buffer
    - ใช้ double buffer จะทำให้ animation ดูลื่นไหลกว่า
  - เลือกว่าจะให้มี buffer อื่นๆ นอกจาก buffer สีอะไรบ้าง
    - ปกติจะใช้แค่ depth buffer สำหรับเก็บความลึกของจุดแต่ละจุด
  - ค่า mode เกิดจากการเอาค่าคงที่ของตัวเลือกต่างๆ มา or กัน
    - ปกติเราจะใช้ GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH
    - กรณีของ code ตัวอย่างใช้ GLUT_SINGLE | GLUT_RGBA

คำสั่งสำหรับจัดการวินโดวส์

- glutCreateWindow(char *string)
  - สร้างวินโดวที่มี title เป็น string ที่ให้
- glutInitWindowPosition(int x, int y)
  - กำหนดตำแหน่งขอบบนของวินโดว
- glutInitWindowSize(int width, int height)
  - กำหนดขนาดของวินโดว

glutDisplayFunc

- glutDisplayFunc(void (*func)(void))
  - กำหนดฟังก์ชันที่ GLUT จะเรียกทุกครั้งเมื่อมันต้องวาดหน้าจอใหม่
  - ฟังก์ชันที่จะส่งให้ glutDisplayFunc ต้องมี prototype
    void <ชื่อฟังก์ชัน>(void)
  - ยกตัวอย่างเช่นฟังก์ชัน void draw() ในโค้ดตัวอย่าง
  - ฟังก์ชันนี้ส่วนมากจะเต็มไปด้วยคำสั่ง OpenGL

glutMainLoop

- glutMainLoop()
  - ฟังก์ชันสุดท้ายที่เราเรียกในโปรแกรม
  - สั่งให้ GLUT ไปทำงานของมัน
  - งานของ GLUT
    - รับ input จากผู้ใช้
    - เรียกฟังก์ชันที่ให้ใน glutDisplayFunc
    - เริ่มต้นใหม่อีกครั้ง
  - ระวัง: ต้องสร้าง windows และกำหนด displayFunc ให้เรียบร้อยก่อนเรียก glutMainLoop