ประมุข ขันเงิน
pramook@gmail.com

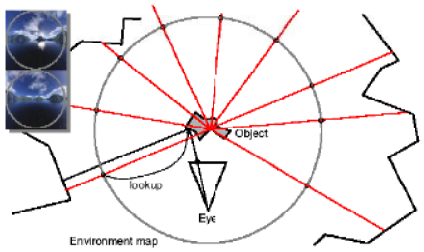
418341 สภาพแวดล้อมการทำงานคอมพิวเตอร์กราฟฟิกส์

การบรรยายครั้งที่ 21

Environment Map

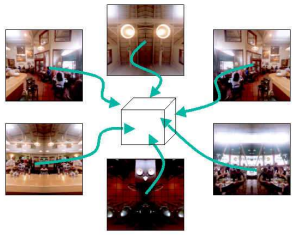
- การใช้ texture เก็บแสงที่พุ่งจาก "สิ่งแวดล้อม" เข้าหาวัตถุในทิศทางต่างๆ
- สมมติว่าวัตถุเป็น "จุด"
- Texture ใช้ในการตอบคำถามว่า "แสงที่พุ่งเข้าหาวัตถุในทิศทาง (x,y,z) มีสีอะไร"

Environment Map



Cube Map

- เป็นวิธีเก็บ environment map แบบหนึ่ง
- ใช้ภาพหกภาพมาประกอบกันเป็นลูกบาศก์



Cube Map ใน OpenGL

- เราสามารถใช้ cube map ใน OpenGL ได้ตั้งแต่ OpenGL เวอร์ชัน 1.3
- ต้องใช้ extension ชื่อ EXT_texture_cube_map
- หมายความว่าเวลาเขียนโปรแกรมใน Windows จะต้องใช้ GLEW

การสร้าง Cube Map

- มีขั้นตอนคล้ายกับการสร้าง texture ธรรมดา
  - Enable การใช้ cube map
  - สร้าง handle ของ cube map ด้วย glGenTextures
  - ทำการ bind cube map ที่สร้างขึ้น
  - Download รูปที่ใช้ทำ cube map ลงสู่ GPU

Enable การใช้ Cube Map

- ให้สั่ง

```
glEnable(GL_TEXTURE_CUBE_MAP_EXT);
```

- และอย่าลืมสั่ง

```
glDisable(GL_TEXTURE_CUBE_MAP_EXT);
```

ก่อนใช้งาน texture แบบอื่น

การสร้าง Handle ของ Cube Map

- เช่นเดียวกับการสร้าง texture อื่นเราต้องประกาศตัวแปรประเภท GLunit เพื่อใช้เก็บชื่อของ cube map

```
GLuint cubeMap;
```

- หลังจากนั้นใช้ glGenTextures สร้าง texture ตามปกติ

```
glGenTextures(1, &cubeMap);
```

Bind Cube Map ที่สร้างขึ้น

- สั่ง glBindTexture โดยใช้ target เป็น GL_TEXTURE_CUBE_MAP_EXT

```
glBindTexture(GL_TEXTURE_CUBE_MAP_EXT,
   cubeMap)
```

Download รูป

- ใช้คำสั่ง glTexImage2D หรือ gluBuild2DMipmaps เพื่อ download รูปเช่นเดิม แต่เราต้อง download รูปจำนวนทั้งหมด 6 รูปสำหรับ 6 ด้านของลูกบาศก์
- เราสามารถระบุว่าจะ download รูปของด้านไหนได้ด้วยการระบุ target ของคำสั่งทั้งสองดังในสไลด์หน้าต่อไป

Target สำหรับ Download รูป

Target	ด้าน
GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT	ขวา
GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT	ซ้าย
GL_TEXTURE_CUBE_MAP_POSITIVE_Y_EXT	บน
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_EXT	ล่าง
GL_TEXTURE_CUBE_MAP_POSITIVE_Z_EXT	หน้า
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_EXT	หลัง

ตัวอย่างโค้ด

- ผมสร้างฟังก์ชัน loadCubeMapSide ไว้สำหรับ download รูปเข้าไปยังด้านหนึ่งของ cube map โดยกำหนด
  - Target ที่จะ download รูปลงไป
  - ชื่อไฟล์ของรูปนั้น
- loadCubeMapSide ใช้ DevIL ในการดึงข้อมูลรูปออกมาจากไฟล์

loadCubeMapSide

```
void loadCubeMapSide(GLuint target,
  const char *imageName)
{
  ILuint image;
  ilGenImages(1, &image);
  ilBindImage(image);
  ilLoadImage((wchar_t *)imageName);
  ilConvertImage(IL_RGB, IL_UNSIGNED_BYTE)
  gluBuild2DMipmaps(target,
     ilGetInteger(IL_IMAGE_BPP),
     ilGetInteger(IL_IMAGE_WIDTH),
     ilGetInteger(IL_IMAGE_HEIGHT),
     ilGetInteger(IL_IMAGE_FORMAT),
     GL_UNSIGNED_BYTE,
     ilGetData());
  ilDeleteImages(1, &image);
}
```

โค้ดตัวอย่าง

- ผมเขียนฟังก์ชัน initCubeMap เพื่อ load รูปทั้งสำหรับทั้ง 6 ด้าน
- ใน initCubeMap ผมเรียก loadCubeMapSide เป็นจำนวนหกครั้งเพื่อ download รูป

initCubeMap

```
void initCubeMap()
{
   glEnable(GL_TEXTURE_CUBE_MAP_EXT);
   glGenTextures(1, &cubeMap);
   glBindTexture(GL_TEXTURE_CUBE_MAP_EXT, cubeMap);

   glTexParameteri(GL_TEXTURE_CUBE_MAP_EXT, GL_TEXTURE_WRAP_S, GL_CLAMP);
   glTexParameteri(GL_TEXTURE_CUBE_MAP_EXT, GL_TEXTURE_WRAP_T, GL_CLAMP);
   glTexParameteri(GL_TEXTURE_CUBE_MAP_EXT, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_CUBE_MAP_EXT, GL_TEXTURE_MIN_FILTER,  GL_LINEAR_MIPMAP_LINEAR);

   loadCubeMapSide(GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT, "../images/cm_right.jpg");
   loadCubeMapSide(GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT, "../images/cm_left.jpg");
   loadCubeMapSide(GL_TEXTURE_CUBE_MAP_POSITIVE_Y_EXT, "../images/cm_top.jpg");
   loadCubeMapSide(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_EXT, "../images/cm_bottom.jpg");
   loadCubeMapSide(GL_TEXTURE_CUBE_MAP_POSITIVE_Z_EXT, "../images/cm_front.jpg");
   loadCubeMapSide(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_EXT, "../images/cm_back.jpg");

   glDisable(GL_TEXTURE_CUBE_MAP_EXT);
}
```

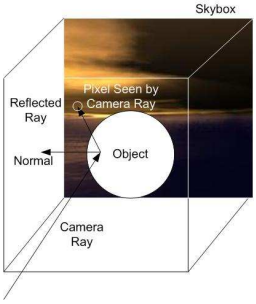
การนำ Cube Map ไปใช้งาน

- กระจก = การสะท้อนแสง
- แก้วใส = การหักเหแสง

หลักการสร้างกระจก

- สร้าง "skybox" หรือกล่องที่มีรูปท้องฟ้า ล้อมรอบวัตถุไว้
- สำหรับ fragment แต่ละ fragment ให้คำนวณทิศทางที่แสงที่เดินทางจากตาไปยังตำแหน่งของ fragment แล้วสะท้อนออกไป
- ให้นำทิศทางที่ได้ไปอ่านข้อมูลจาก cube map

หลักการสร้างกระจก



การสร้างกระจกใน OpenGL

- โดยปกติแล้วเวลาจะอ่านข้อมูลจาก cube map เราจะต้องใช้ texture coordinate 3 ตัว เนื่องจากทิศทางเป็นทิศทางในสามมิติ
- เราสามารถกำหนดทิศทางได้เองด้วยคำสั่ง glTexCoord3d
- แต่ละ component ของทิศทางจะมีค่าตั้งแต่ -1 ถึง 1
  - ไม่ใช้ 0 ถึง 1 เหมือนกับ texture coordinate อื่นๆ

การสร้างกระจกใน OpenGL

- Texture coordinate ที่จะใช้มีสามตัวคือ s, t, และ r
- เราสามารถสั่งให้ OpenGL สร้าง texture coordinate ให้โดยอัตโนมัติได้ด้วยคำสั่ง glEnable(GL_TEXTURE_GEN_?)
  - ถ้าอยากให้สร้าง s ให้โดยอัตโนมัติก็สั่ง
    glEnable(GL_TEXTURE_GEN_S);
  - ถ้าอยากให้สร้าง t ให้โดยอัตโนมัติก็สั่ง
    glEnable(GL_TEXTURE_GEN_T);

การสร้างกระจกใน OpenGL

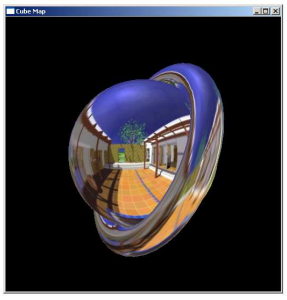
- นอกจากนี้ยังต้องบอกด้วยว่าจะให้สร้าง texture coordinate ให้แบบใด ด้วยคำสั่ง glTexGeni
- ในกรณีการสร้างกระจกเราต้องสั่ง

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
  GL_REFLECTION_MAP_EXT);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE,
  GL_REFLECTION_MAP_EXT);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE,
  GL_REFLECTION_MAP_EXT);
```

ตัวอย่างโค้ด

```
void display()
{
  glEnable(GL_TEXTURE_CUBE_MAP_EXT);

  glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
     GL_REFLECTION_MAP_EXT);
  glTexGeni(GL_T, GL_TEXTURE_GEN_MODE,
     GL_REFLECTION_MAP_EXT);
  glTexGeni(GL_R, GL_TEXTURE_GEN_MODE,
     GL_REFLECTION_MAP_EXT);

  glEnable(GL_TEXTURE_GEN_S);
  glEnable(GL_TEXTURE_GEN_T);
  glEnable(GL_TEXTURE_GEN_R);

  glutSolidSphere(1.5, 50, 50);

  glutSwapBuffers();
}
```

ดู demo



การใช้ Cube Map ใน Cg

- Cg ให้ผู้ใช้สามารถประกาศ uniform parameter ประเภท samplerCUBE ใน fragment profile ได้ เช่น

```
void main(
  float3 worldPosition : TEXCOORD0,
  float3 worldNormal : TEXCOORD1,
  uniform samplerCUBE env,
  uniform float3 eyePosition,
  out float4 color : COLOR)
{
    :
    :
}
```

การใช้ Cube Map ใน Cg

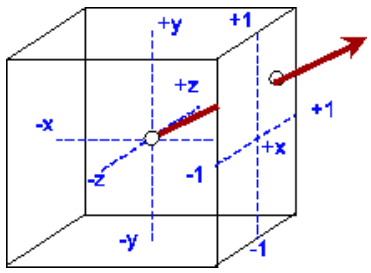
- เวลาอ่านข้อมูลจาก cube map ให้ใช้คำสั่ง texCUBE โดย
  - Parameter ตัวแรกเป็นตัวแปรประเภท samplerCUBE
  - Parameter ตัวที่สองเป็นค่าประเภท float3
- ตัวอย่าง

```
color = texCUBE(env, float3(1,0,0));
```

การใช้ Cube Map ใน Cg

- ตัวแปรประเภท float3 ที่เราให้ไปต้องเป็นเวกเตอร์หนึ่งหน่วยที่แต่ละมิติมีค่าอยู่ในช่วง [-1,1]
- ค่าที่อ่านได้จาก cube map คือค่าของสีที่จุดที่เกิดจากการยิงรังสีจากจุด (0,0,0) ไปในทิศทางที่กำหนดด้วยตัวแปร float3 ที่ให้ฟังก์ชัน texCUBE ไปตัดกับกล่องลูกบาศก์ที่มีความยาวด้านละสองหน่วยที่มีจุดศูนย์กลางอยู่ที่จุด (0,0,0)

การใช้ Cube Map ใน Cg



การทำกระจกใน Cg

- ให้ vertex program คำนวณ
  - ตำแหน่งใน world space ของแต่ fragment
  - Normal ใน world space แต่ละ fragment
- ให้ fragment program รับตำแหน่งของตา
- แล้วให้ fragment program คำนวณ
  - เวกเตอร์ทิศทางการสะท้อนแสงซึ่งเกิดจากแสงจากตา เดินทางไปยังตำแหน่งของ fragment
  - เอาเวกเตอร์ทิศทางที่ได้ไปอ่านสีจาก cube map

Vertex Program สำหรับทำกระจก

```
void main(float4 inPosition : POSITION,
  float3 normal : NORMAL,
  out float4 clipPosition : POSITION,
  out float3 worldPosition : TEXCOORD0,
  out float3 worldNormal : TEXCOORD1,
  uniform float4x4 mvp : state.matrix.mvp,
  uniform float4x4 mv,
  uniform float4x4 mvit)
{
  clipPosition = mul(mvp, inPosition);
  worldPosition = mul(mv, inPosition).xyz;
  worldNormal = mul(mvit, float4(normal,
  0)).xyz;
}
```

Vertex Program สำหรับทำกระจก

- ความหมายของตัวแปร
  - mv ใช้เก็บ modeling matrix
  - mvit ใช้เก็บ inverse transpose ของ modeling matrix
- ตัวแปรสองตัวแปรข้างต้นผู้ใช้จะต้องเป็นคนกำหนดเอง เนื่องจากใน OpenGL มันจะรวม modeling matrix กับ viewing matrix เข้าด้วยกัน

Vertex Program สำหรับทำกระจก

- สังเกตว่าเราใช้ตัวแปร worldPosition ซึ่งมี semantic เป็น TEXCOORD0 สำหรับเก็บ world position ของ vertex
- และใช้ worldNormal ซึ่งมี semantic เป็น TEXCOORD1 สำหรับเก็บ normal ใน world space ของแต่ละ vertex
- ที่ต้องทำเช่นนี้เพราะ output ของ vertex program ไม่มี semantic สำหรับเก็บ normal หรือตำแหน่งใน world space

Fragment Program สำหรับทำกระจก

```
void main(
    float3 worldPosition : TEXCOORD0,
    float3 worldNormal : TEXCOORD1,
    uniform samplerCUBE env,
    uniform float3 eyePosition,
    out float4 color : COLOR)
{
  worldNormal = normalize(worldNormal);
  float3 eyeToP = normalize(worldPosition
        - eyePosition);
  float3 reflected = reflect(eyeToP,
        worldNormal);
  color = texCUBE(env, reflected);
}
```

Fragment Program สำหรับทำกระจก

- เราทำการคำนวณเวกเตอร์ทิศทางของแสงที่เดินทางจากตาไปยังตำแหน่งของ fragment ด้วยคำสั่ง

```
float3 eyeToP = normalize(worldPosition -
   eyePosition);
```

- หลังจากนั้นคำนวณทิศทางที่แสงสะท้อนออกไปด้วยคำสั่ง

```
float3 reflected = reflect(eyeToP,
   worldNormal);
```

- ฟังก์ชัน reflect มีไว้สำหรับคำนวณเวกเตอร์แสงสะท้อน

Fragment Program สำหรับทำกระจก

- ขั้นสุดท้าย เรานำเอาทิศทางของเวกเตอร์แสงสะท้อนไปอ่านค่าจาก cube map

```
color = texCUBE(env, reflected);
```

ดู demo

การทำวัตถุผิวมันวาว

- เราอาจมองว่าพื้นผิวของวัตถุมันวาวมีส่วนประกอบอยู่สองส่วน
  - ส่วนหนึ่งมีพฤติกรรมเหมือนกระจก
  - อีกส่วนมีพฤติกรรมตาม lighting model อื่น เช่น phong lighting model
- สีของพื้นผิวประเภทนี้เกิดจากการนำเอาสีที่ได้จากการสะท้อนแสงแบบกระจกมารวมกับสีที่ได้จากพฤติกรรมการสะท้อนแสงอื่นๆ

การทำวัตถุผิวมันวาว

- เราสามารถคำนวณสีของทั้งสองส่วน
  - สมมติว่าส่วนแรกได้สี a และ
  - ส่วนที่สองได้สี b
- เรากำหนดเลข w (ย่อคำว่า weight) โดยที่ $0 \le w \le 1$ แล้วให้สีขั้นสุดท้ายมีค่าเท่ากับ

$$color = w \times a + (1-w) \times b$$

ดู demo