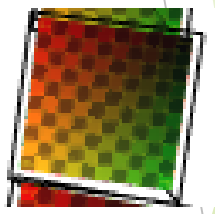
繪製功能 / 基本功能

- 畫出有一個有顏色的正方形（有外框）

由給定的座標乘上視角的反矩陣和投影，
將每個vertex畫出來

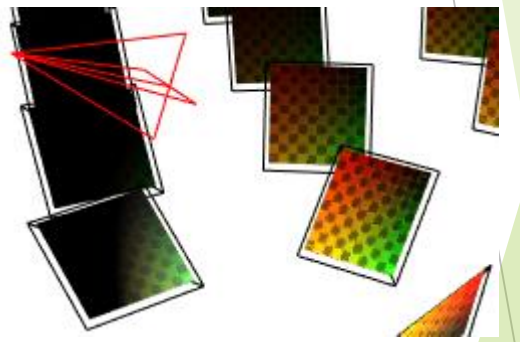


```
function drawPlane(matrix, viewProjectionMatrix, matrixLocation, numVertices) {
  matrix = m4.multiply(viewProjectionMatrix, matrix);
  gl.uniformMatrix4fv(matrixLocation, false, matrix);
  var primitiveType = gl.LINES;
  var offset = 0;
  gl.drawArrays(primitiveType, offset, numVertices);
}
```

繪製功能 / 基本功能

- 畫出15x15個正方形平板 &
  正方形可以根據投射燈位置改變朝向

透過15*15迴圈算出各正方形位置，
再透過lookAt看向投射燈



```
var deep = 15;
var across = 15;
for (var zz = 0; zz < deep; ++zz) {
  var v = zz / (deep - 1);
  var z = (v - .5) * deep * 150;
  for (var xx = 0; xx < across; ++xx) {
    var u = xx / (across - 1);
    var x = (u - .5) * across * 150;
    var matrix = m4.lookAt([x, 0, z], target, up);
    drawPlane(matrix, viewProjectionMatrix, matrixLocation, numVertices);
  }
}
```

繪製功能 / 基本功能

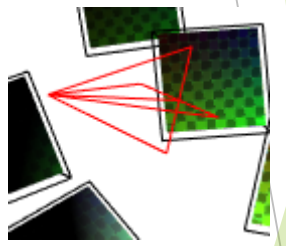
- 畫出投射燈&永遠照著世界中心

因解說投影片中出現線段和平面兩種投射燈

這裡選擇出現最多次的線段實作

投射燈從斜上方照下，永遠照往世界中心點，

顏色永遠為暗紅色（不受光線影響）



```
function drawSpotlight(matrix, viewProjectionMatrix, matrixLocation, numSL) {
  matrix = m4.lookAt(target, [0,0,0], [0,1,0]);
  matrix = m4.multiply(viewProjectionMatrix, matrix);
  matrix = m4.scale(matrix, 3, 3, 3);
  gl.uniformMatrix4fv(matrixLocation, false, matrix);
  var primitiveType = gl.LINES;
  var offset = 10;
  gl.drawArrays(primitiveType, offset, 18);
}
```

繪製功能 / 基本功能

- 投射燈可以改變位置

透過ui更新投射燈的水平和垂直角度


```
function updateTargetRadius(event, ui) {
  targetRadius = ui.value;
  target[0] = Math.sin(targetAngleRadians) * targetRadius;
  target[2] = Math.cos(targetAngleRadians) * targetRadius;
  requestAnimationFrame(drawScene);
  requestAnimationFrame(drawTexture);
}

function updateTargetHeight(event, ui) {
  target[1] = ui.value;
  requestAnimationFrame(drawScene);
  requestAnimationFrame(drawTexture);
}
```

繪製功能 / 基本功能

- 投射燈可以改變位置

透過ui更新投射燈的水平和垂直角度


```
function updateTargetRadius(event, ui) {
  targetRadius = ui.value;
  target[0] = Math.sin(targetAngleRadians) * targetRadius;
  target[2] = Math.cos(targetAngleRadians) * targetRadius;
  requestAnimationFrame(drawScene);
  requestAnimationFrame(drawTexture);
}

function updateTargetHeight(event, ui) {
  target[1] = ui.value;
  requestAnimationFrame(drawScene);
  requestAnimationFrame(drawTexture);
}
```

上色與打光

- 正方形有貼圖

貼圖位置和正方形一樣

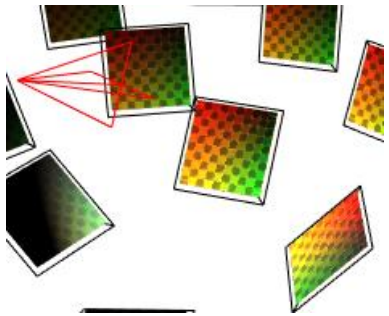
貼圖座標正面由兩組順時針給定，

背面由兩組逆時針三角形給定

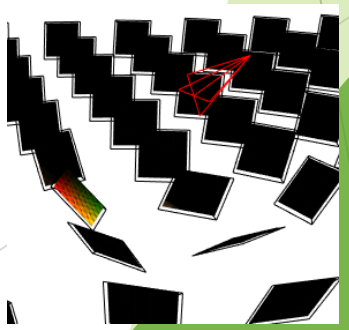
```
var texcoords = [
  //正面
  1, 1, 1, 0, 0, 1,
  0, 0, 0, 1, 1, 0,

  //反面
  1, 1, 0, 1, 1, 0,
  0, 0, 1, 0, 0, 1,
];
```

正面



反面



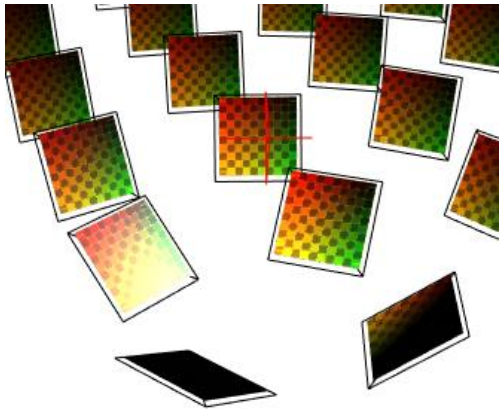
上色與打光

- 正方形貼圖可受到打光影響

最終顏色由2D取樣器從貼圖中取出

乘上光線強弱和光線顏色

加上specular(預設白光)



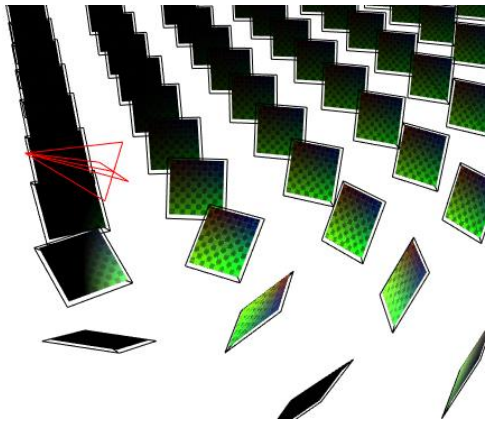
```
outTexture = texture(u_texture, v_texcoord);
outTexture.rgb *= light * u_lightColor;
outTexture.rgb += specular * u_specularColor;
```

上色與打光

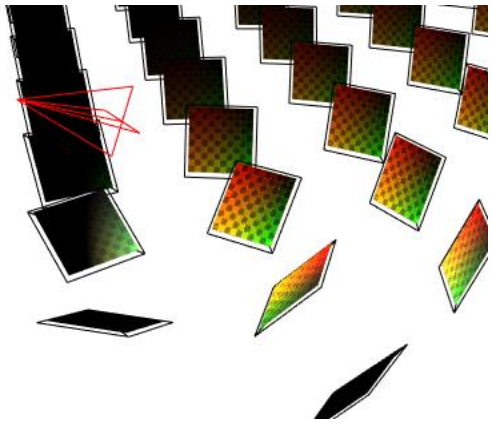
- 有可調整的打光效果

Hue透過HSV轉乘RGB

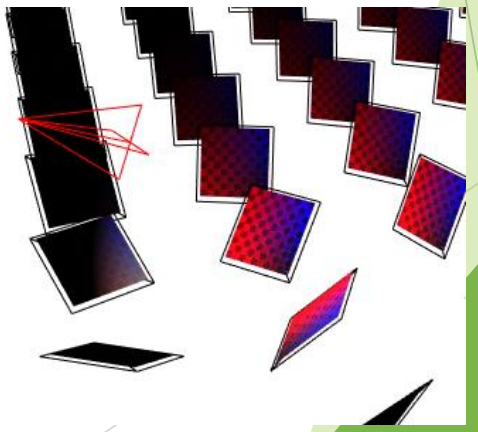
Hue:125



Hue:200



Hue:300

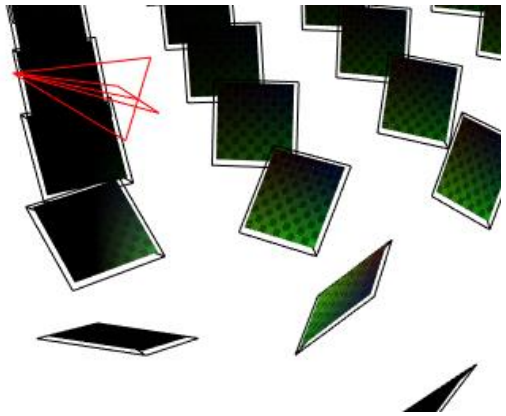


上色與打光

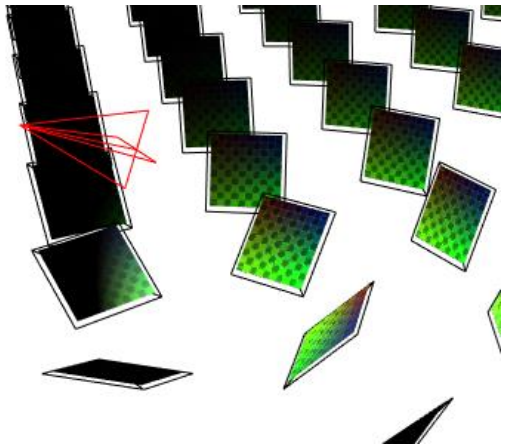
- 有可調整的打光效果

Brightness數值越大，色彩越繽紛

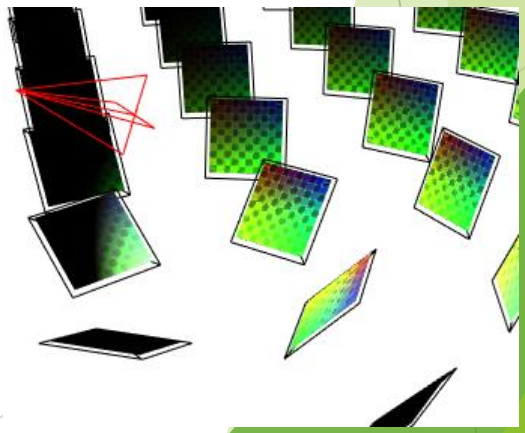
Brightness :10



Brightness :150



Brightness :255

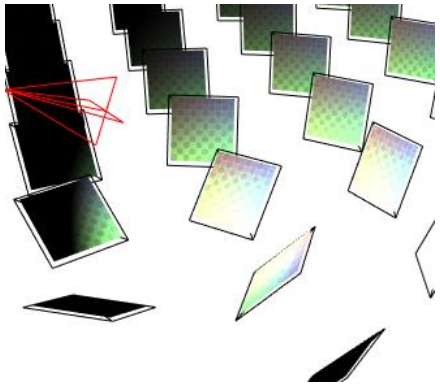


上色與打光

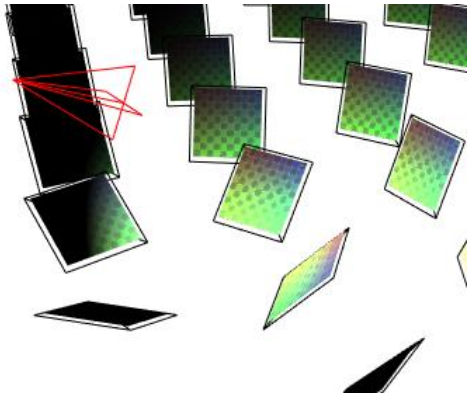
- 有可調整的打光效果

Shininess數值越小，白光越強烈

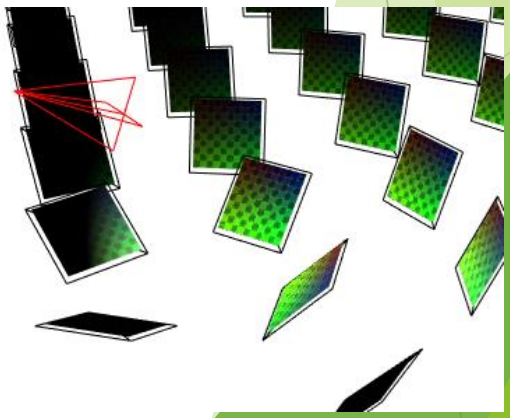
Shininess :2



Shininess :30



Shininess :128



上色與打光

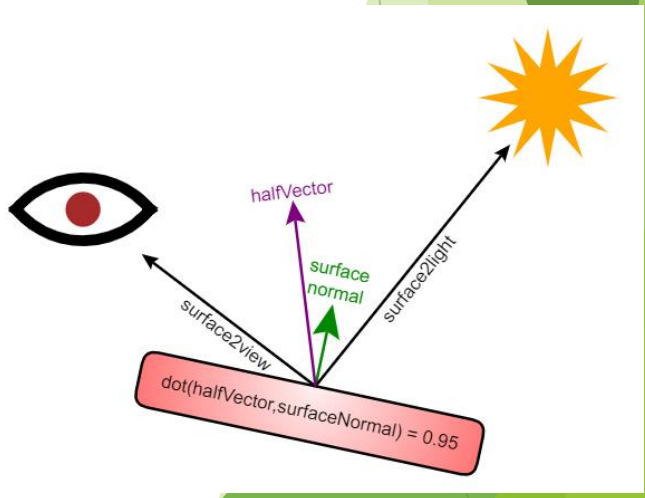
- 打光時有正確運用法向量、光線位置、人的位置（需為Phone Shading）

亮度：

先視角和物體到光源內積算出半角，

算出半角將視角、光源、物體移動到對應的世界座標，

再由物體到後再和物體法向量內積



上色與打光

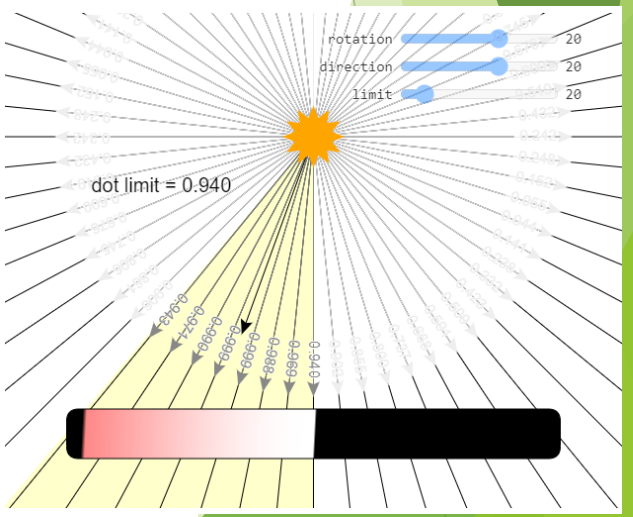
- 打光範圍採用Spotlight的方式

範圍：

由inner angle和outter angle算出光線限制範圍

再考慮和光線方向關係，

透過clamp將範圍以外的截掉



操作

- 上述操作都有對應之UI捲軸
- 可用介面上之捲軸旋轉場景