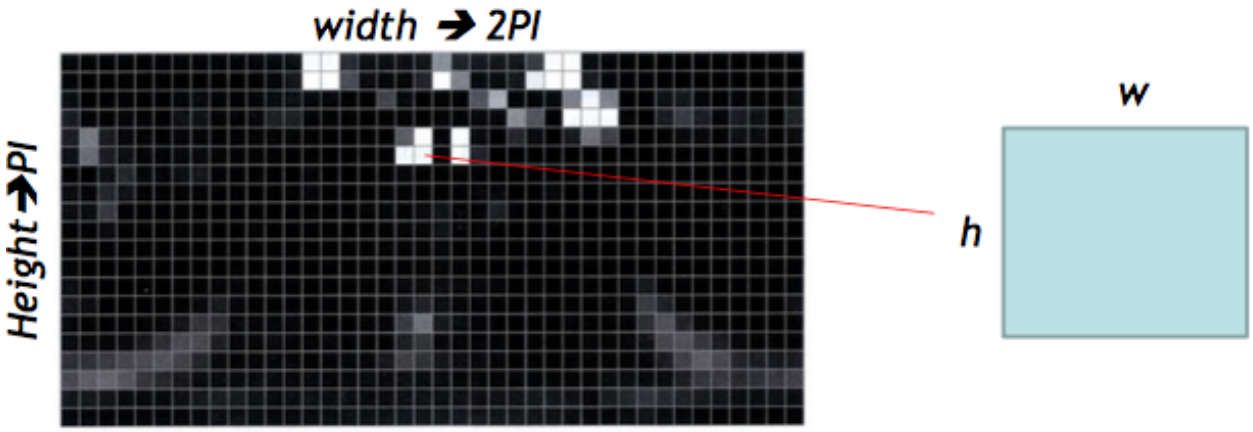
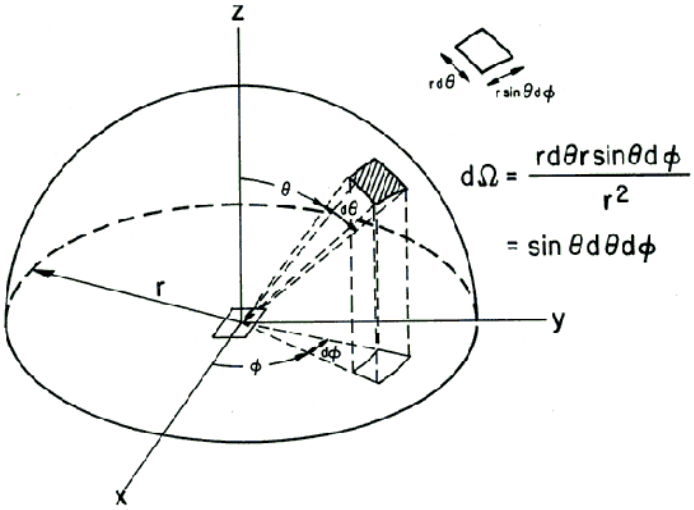
Rendering HW3

資工所 - R04922078 - 吳德彥

Scale the light intensity with the areas (solid angles):





$$d\omega = \sin\theta d\theta d\phi$$

$$p(\theta,\phi)d\theta d\phi = p(\omega)d\omega$$

$$p(\theta,\phi) = \sin\theta p(\omega)$$

```
float solidAngle = ((2.f * M_PI) / (width - 1)) * (M_PI /(height - 1));
texels[u+v*width] *= (solidAngle * sinTheta);
```

- According to the density of sphere in chap 13, the solid angle is calculated from the width and height of texmap.
- Scale the light intensity with the density of area

Summed area table:

According to the paper, A Median Cut Algorithm for Light Probe Sampling, calculating the total energy within regions of the image can be accelerated using a summed area table.

Original Image

3	8	2	1
6	3	9	7
5	2	4	9
6	0	1	8

Sum area table

3	11	13	14
9	20	31	39
14	27	A 42	59 B
20	33	49 D	74 C

Calculate the energy:

```
float *img = new float[width*height];
    for (int v = 0; v < height; ++v) {
        float vp = (float)v / (float)height;
        float sinTheta = sinf(M_PI * float(v+.5f)/float(height));
        for (int u = 0; u < width; ++u) {
            float up = (float)u / (float)width;
            img[u+v*width] = radianceMap->Lookup(up, vp, filter).y();
            img[u+v*width] *= sinTheta;
        }
    }
```

Spectrum.y():

float y() const {
const float YWeight[3] = { 0.212671f, 0.715160f, 0.072169f };
*return YWeight[0] * c[0] + YWeight[1] * c[1] + YWeight[2] * c[2];*
} — from Y = 0.2125R + 0.7154G + 0.0721B following ITU-R Recom- mendation BT.709

Calculate sum area table:

```
    float *sumAreaTable = new float[width*height];
    for (int v = 0; v < height; v++) {
        float sum = 0;
        for (int u = 0; u < width; u++) {
            sum += img[u+v*width] * solidAngle;
            if (v > 0)
                sumAreaTable[u+v*width] = sumAreaTable[u+(v-1)*width] + sum;
            else
                sumAreaTable[u+v*width] = sum;
        }
    }
```

Region Structure:

```
struct Region
{
    int left, top, w, h;
    Region(int left, int top, int width = 0, int height = 0):
        left(left), top(top), w(width), h(height){}
    float getEnergy(float sumAreaTable[], int width, int height)
    {
        float rtotal = sumAreaTable[left + w + (top + h)*width];
        if(left > 0)
            rtotal -= sumAreaTable[left - 1 + (top + h)*width];
        if(top > 0)
            rtotal -= sumAreaTable[left + w + (top - 1)*width];
        if(left > 0 && top > 0)
            rtotal += sumAreaTable[left - 1 + (top - 1)*width];

        return rtotal;
    }
```

Partition:

- Add the entire light probe image to the region list as a single region.
- For each region in the list, subdivide along the longest dimension such that its light energy is divided evenly.
  - Find the half of total energy
  - Find the region larger than half of total energy
  - Move in pixels to find the evenly region
- if the number of iterations is less than n, return to step2.

```
while(regions.size() < n)
{
    printf("iteration:  %lu\n", regions.size());
    std::vector<Region> nextRegions;
    for (vector<Region>::iterator it = regions.begin(); it != regions.end(); ++it){

        Region region = *it;
        int offset = 0;
        int dir = 1;
        float targetEnergy = region.getEnergy(sumAreaTable, width, height) * 0.5f;
        // int cutAxis = region.getCutAxis();
        Region r = region.getRegionFromMidPoint(offset, width, height);

        int deltaOffset = region.getMidPoint();
        while(deltaOffset >= 1)
        {
            float energy = r.getEnergy(sumAreaTable, width, height);
            if(energy > targetEnergy )
                offset -= (deltaOffset/2 + deltaOffset % 2);
            else if (energy < targetEnergy)
                offset += (deltaOffset/2 + deltaOffset % 2);
            else
                break;
            deltaOffset /= 2;
            r = region.getRegionFromMidPoint(offset, width, height);
        }


        nextRegions.push_back(r);
        nextRegions.push_back(region.getOtherRegionFromMidPoin(offset, width, heigh
    }
    regions = nextRegions;
}
```

Set light source color:

```
// Assign VPLs
_pdf = 1.f / regions.size();
for (vector<Region>::iterator it = regions.begin(); it != regions.end(); ++it){
    Region region = *it;
    RGBSpectrum spectrum = RGBSpectrum(0.f);
    int index = region.left + region.top * width;
    for(int v = 0; v <= region.h; ++v){
        for(int u = 0; u <= region.w; ++u){
            spectrum += texels[index + (u + v * width)];
        }
    }
    _VPLs.push_back(VPL(region.getCenterIndex(width, height), spectrum));
}
```

- Place a light source at the center or centroid of each region, and set the light source color to the sum of pixel values within the region.

The center of region:

```
float* getCenter(float width, float height)
{
    float* center = new float[2];
    center[0] = float(left + w / 2) / width;
    center[1] = float(top + h / 2) / height;
    return  center;
}
```

IsDeltaLight:

bool IsDeltaLight() const { return true; }

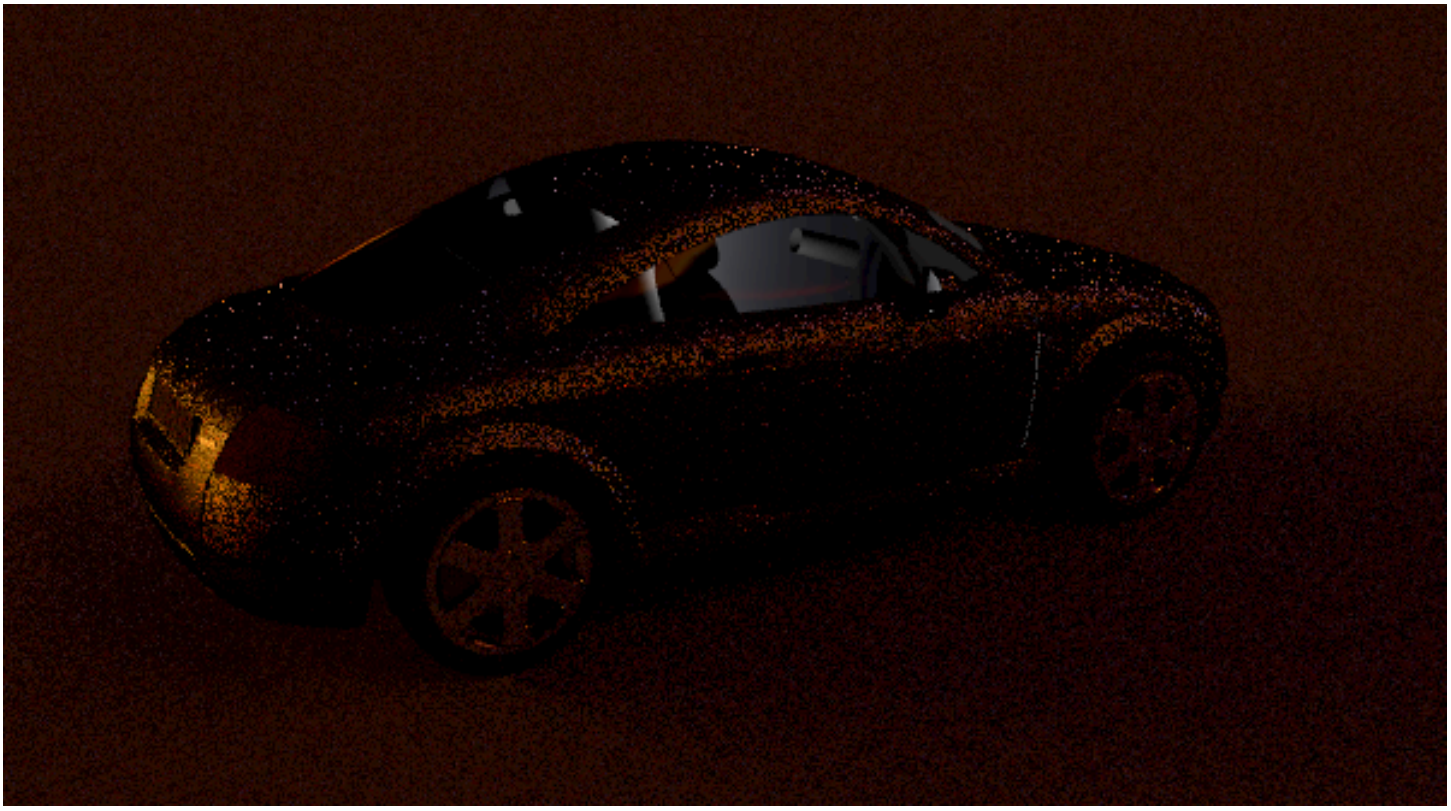
Sample_L:

- We can simply random choose one from n lights and return PDF with 1/n
- return its direction, intensity and PDF

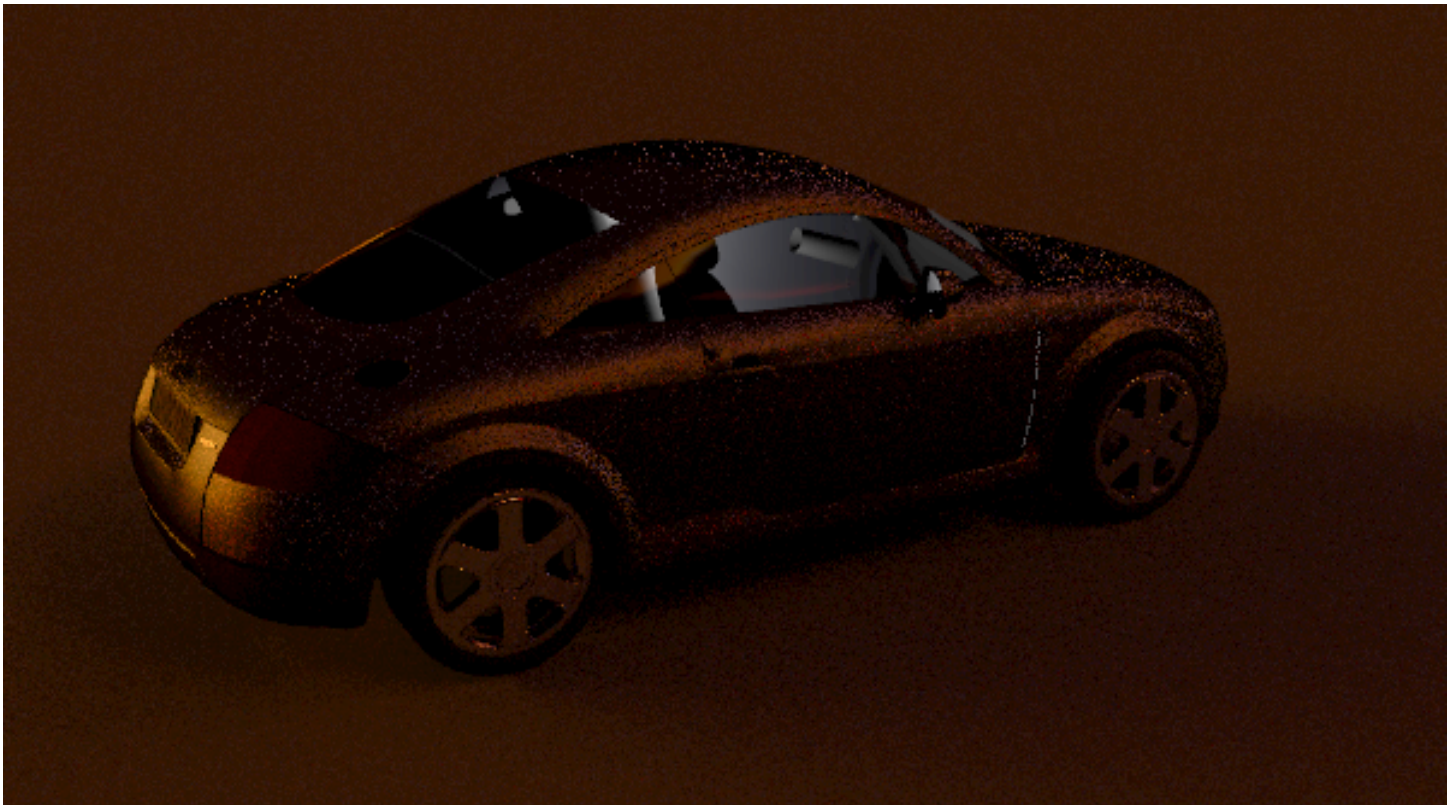
```
Spectrum MedianCutEnvironmentLight::Sample_L(const Point &p, float pEpsilon,
        const LightSample &ls, float time, Vector *wi, float *pdf,
        VisibilityTester *visibility) const {
    PBRT_INFINITE_LIGHT_STARTED_SAMPLE();
    // Find vpl with random
    VPL vpl = VPLs[Floor2Int(ls.uComponent * VPLs.size())];

    // Convert infinite light sample point to direction
    float theta = vpl.Center[1] * M_PI, phi = vpl.Center[0] * 2.f * M_PI;
    float costheta = cosf(theta), sintheta = sinf(theta);
    float sinphi = sinf(phi), cosphi = cosf(phi);
    *wi = LightToWorld(Vector(sintheta * cosphi, sintheta * sinphi,
                              costheta));

    // Compute PDF for sampled infinite light direction
    *pdf = _pdf;

    // Return radiance value for infinite light direction
    visibility->SetRay(p, pEpsilon, *wi, time);
    Spectrum Ls = Spectrum(vpl.Spectrum,
                           SPECTRUM_ILLUMINANT);
    PBRT_INFINITE_LIGHT_FINISHED_SAMPLE();
    return Ls;
}
```

所有結果圖：
Envlight samples

Sample 4 (0.6s)

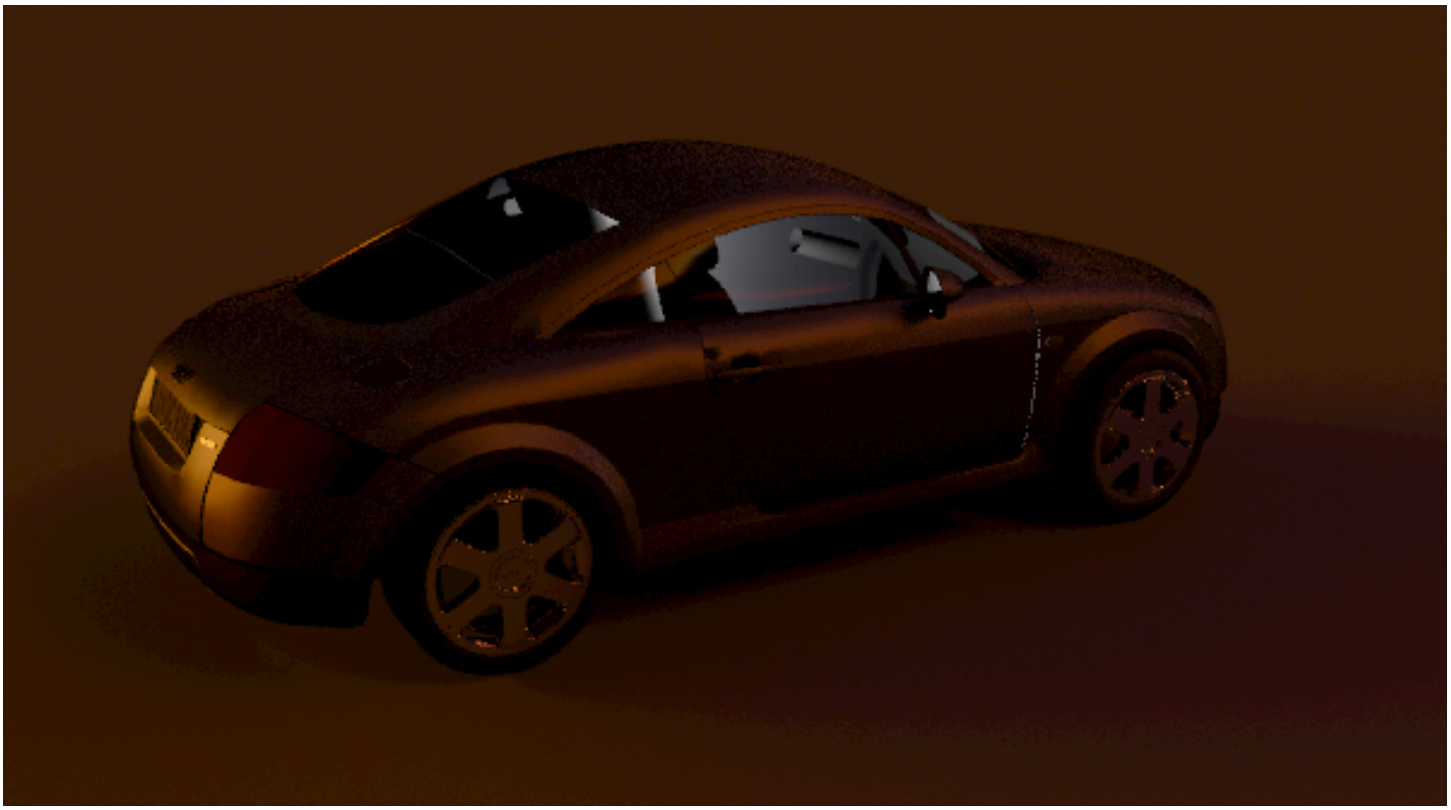


Sample 16 (1.1s)



Sample 64 (3.5s)

Sample 256(12.9s)



My Envlight samples

Sample 4 (1.1s)

Sample 16 (3.2s)

Sample 64 (11.9s)

Sample 256 (49.1s)

New Envlight samples

Sample 4 (0.8s)

Sample 16 (2.2s)

Sample 64 (8.4s)

Sample 256 (32.7s)

My New Envlight samples

Sample 4 (0.5s)

Sample 16 (1.6s)

Sample 64 (5.1s)

Sample 256 (15.7s)

執行環境(Mac.OS)及配置:

Core: 8 cores, 2.2Hz, I7
Memory: 16GB