ICG HW4 Report

Group ID: 16
Theme: The Cosmic Breakdown
Members: 111550113 謝詠晴、111550057 莊婷馨

1. Introduction

1) Storyline:

在寂靜的夜空中，一艘神秘的 UFO 緩緩飛入，散發著濃烈的科技感。此時，UFO 嘗試瞬間移動離開，然而 UFO 卻開始出現異常燈光閃爍，彷彿有壞事要發生。突然間，一個外星人身影出現了，似乎是因 UFO 故障而被迫跳出。

因為接觸到外界宇宙的氣體後，外星人的身體開始發生變異，表皮迅速長出五彩繽紛又忽長忽短的毛髮，但它並未因此停下，而是開始環繞著 UFO，嘗試尋找 UFO 故障的原因。不幸的是，隨著身體的異變逐漸失控，外星人的身體開始裂開，而 UFO 也出現如雜訊般的亂閃。

最終，一聲巨響劃破夜空，UFO 和外星人瞬間爆炸，橘色光芒吞噬一切，留下殘骸與未解的謎團。這場「銀河意外」究竟是什麼？只有夜空知曉答案。

2) Design Process:

a. UFO 飛入場景
- 使用 Gouraud 營造科技感

b. UFO 嘗試瞬間移動
- 利用 Geometry shader 達成周期性的膨脹和收縮

c. UFO 發生異常閃爍
- 不同顏色的燈光變化

d. 外星人現身（長毛效果＋繞 **UFO** 公轉）
- 利用 Geometry shader 實作 Normal Visualization 達到長毛效果

e. 爆炸前徵兆（外星人裂開 **+ UFO** 雜訊亂閃）
- 裂開：利用 Geometry shader 將三角形沿法向量方向平移
- 雜訊亂閃：利用 Geometry shader 在頂點添加隨機偏移

f. 爆炸噴發
- 利用 Geometry shader 繪製 Normal 方向的線條，配合 Fragment shader 呈現漸變顏色，模擬爆炸的效果

2. Implementation Details

1) Object

因為本次作業的重點著重在 Geometry shader，因此我們直接在網路上搜尋可以使用的 .obj 檔案，下載連結參見 References。

2) CubeMap

我們想要找尋外太空的 skybox 照片檔案，但發現有困難，因此參考了 Reddit 的留言回復，到 Nasa 官方網站下載全景的照片，再經由他人寫好的程式將全景照片轉換成立方體貼圖。

3) **Shader**

a. **Gouraud Shading**

同作業三的程式碼

b. **Grow**

- 在 vertex shader 中將 position 和 normal 轉換到 world coordinate
- 在 geometry shader 中將每個 vertex 往 normal 方向向外移動，形成膨脹的效果，膨脹程度由 inflateAmount 控制

```
for (int i = 0; i < 3; ++i) {
    vec3 inflatedPosition = vFragPos[i] + vNormal[i] * inflateAmount;
    gFragPos = inflatedPosition;
    gNormal = vNormal[i];
    gTexCoord = vTexCoord[i];

    gl_Position = projection * view * vec4(inflatedPosition, 1.0);
    EmitVertex();
}
EndPrimitive();
```

- 在 fragment shader 中計算 diffuse 和 specular 的光，加上 texture 顏色，做最終呈現的顏色

```
vec3 norm = normalize(gNormal);
vec3 lightDir = normalize(light.position - gFragPos);
float diff = max(dot(norm, lightDir), 0.0);

vec3 viewDir = normalize(CameraPos - gFragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32.0);

vec3 lightColor = vec3(1.0, 1.0, 1.0);
vec3 result = (1.0 + diff + spec) * lightColor;

vec4 texColor = texture(ourTexture, gTexCoord);
FragColor = vec4(texColor.rgb * result, texColor.a);
```

c. **Alert**

- 在 vertex shader 中把每個 vertex 的 normal 轉換到 world coordinate
- 在 fragment shader 中用了三種 baseColor (紅、橙、黃)，intensity 會隨時間有 sin 函數的變化，再利用 mix 函數根據不同的 intensity 範圍進行顏色混合，同時使用簡單的 random 函數為顏色增加細微的隨機性

```
vec3 fireColor;
if (intensity < 0.33) {
    fireColor = mix(baseColor1, baseColor2, intensity * 3.0);
} else if (intensity < 0.66) {
    fireColor = mix(baseColor2, baseColor3, (intensity - 0.33) * 3.0);
} else {
    fireColor = mix(baseColor3, baseColor1, (intensity - 0.66) * 3.0);
}

fireColor += randomNoise * 0.1;
```

與 diffuse 和 specular light 結合後呈現最後的顏色

d. **Fur**

- 在 vertex shader 中把每個 vertex 的 normal 轉換到 world coordinate
- 在 geometry shader 中對每個 vertex 生成一個新的 vertex，並把兩個 vertex 連接成為 line。兩個 vertex 的距離由 float MAGNITUDE 控制，代表毛的長度。

```
uniform mat4 projection;
uniform float time;
float MAGNITUDE = 1.0f + sin(time) * 0.5f;

void GenerateLine(int index){
    gl_Position = projection * gl_in[index].gl_Position;
    EmitVertex();
    gl_Position = projection * (gl_in[index].gl_Position + vec4(gs_in[index].normal, 0.0) * MAGNITUDE);
    EmitVertex();
    EndPrimitive();
}
```

- fragment shader 中用 sin 函數控制毛的顏色，呈現不斷變化的效果

```
float r = abs(sin(time * 1.0));
float g = abs(sin(time * 0.5 + 1.0));
float b = abs(sin(time * 0.25 + 2.0));
FragColor = vec4(r, g, b, 1.0f);
```

e. Split

- 在 geometry shader 中，使用 triangle 的邊做 cross product，得到 surface normal

```
vec3 GetNormal(){
   vec3 a = vec3(gl_in[0].gl_Position) - vec3(gl_in[1].gl_Position);
   vec3 b = vec3(gl_in[2].gl_Position) - vec3(gl_in[1].gl_Position);
   return normalize(cross(b, a));
}
```

設定 split() 函數，會 return 某個 position 沿著特定方向前進特定距離後的位置

```
vec4 split(vec4 position, vec3 normal, float offset){
    vec3 direction = normal * offset;
    return position + vec4(direction, 0.0);
}
```

分別沿著 normal 和 -normal 的方向前進 offsetMagnitude 的距離，得到兩個 triangle，offsetMagnitude 由 sin 函數控制

```
float offsetMagnitude = ((sin(time) + 1.0) / 2.0) * 2.0;
for(int i = 0; i < 3; i++){
    gl_Position = explode(gl_in[i].gl_Position, normal, offsetMagnitude);
    TexCoords = gs_in[i].TexCoord;
    EmitVertex();
}
EndPrimitive();

for(int i = 0; i < 3; i++){
    gl_Position = explode(gl_in[i].gl_Position, -normal, offsetMagnitude);
    TexCoords = gs_in[i].TexCoord;
    EmitVertex();
}
EndPrimitive();
```

- fragment shader 根據 TexCoords 拿取 texture 的顏色並顯示

f. Noise

- 在 vertex shader 中將 position 和 normal 轉換到 world coordinate
- 以 triangle 的形式 input 到 geometry shader，針對 position 生成 randomOffset，範圍介於 -1 和 1 之間，產生雜訊的效果

```
vec4 noise(vec4 position){
    vec3 randomOffset = vec3(
        rand(position.xyz) - 0.5,
        rand(position.yzx) - 0.5,
        rand(position.zxy) - 0.5
    ) * 2.0;
    return position + vec4(randomOffset.x, randomOffset.y, randomOffset.z, 0.0);
}
```

將加上 noise 之後的 position 轉換到 projected coordinate

```
void main() {
    for(int i = 0; i < 3; i++){
        gl_Position = projection * view * noise(gl_in[i].gl_Position);
        TexCoords = gs_in[i].TexCoord;
        EmitVertex();
    }
    EndPrimitive();
}
```

- fragment shader 根據 TexCoords 拿取 texture 的顏色並顯示

g. Explosion

- 在 main.cpp 中的 update() 函式更新 float explode 的值，隨著 render 次數增加往外噴射的速度，並作為 uniform 傳入 geometry shader

```
increase += 0.04;
explode = explode + increase;
```

- 在 vertex shader 計算 world coordinate 的 position
- 以 triangle 的形式 input 到 geometry shader，並計算平面的 normal

```
vec3 vector0 = vec3(gl_in[0].gl_Position - gl_in[1].gl_Position);
vec3 vector1 = vec3(gl_in[2].gl_Position - gl_in[1].gl_Position);
vec4 surfaceNormal = vec4(normalize(cross(vector0, vector1)), 0.0f);
```

針對每個傳入的 vertex 計算出 2 個新的 vertex，都是從原本的位置往 normal 方向前進，前進距離根據 explode 和 start 兩個浮點數決定，並把兩個 vertex 連接成為 line

```
// start pos
gl_Position = projection * (gl_in[i].gl_Position + start * surfaceNormal);
blastColor = getExplosionColor(start);
EmitVertex();

// explode pos
gl_Position = projection * (gl_in[i].gl_Position + explode * surfaceNormal);
blastColor = getExplosionColor(explode);
EmitVertex();

EndPrimitive();
```

vertex 的顏色由前進距離決定，距離近接近橘色，距離遠接近深灰色，模仿爆炸的效果

```
vec4 getExplosionColor(float distance) {
    float ratio = min(distance / 30, 1.0);
    vec3 color = mix(vec3(1.0, 0.5, 0.1), vec3(0.1, 0.1, 0.1), ratio);

    return vec4(color, 1.0);
}
```

- fragment shader 直接輸出 interpolation 過後的顏色

3. Discussion

1) Role Playing

Proposer 起初建議以類似《玩具總動員》的玩具物體作為主角，但經討論後，Critic 認為應優先確保能找到合適的 obj 檔案，再決定主題。Proposer 提議使用 Blender 自行繪製物體，但 Critic 擔心學習 Blender 可能耗時且偏離作業重點。最終，雙方決定採用網路上現成的 obj 檔案。

由於 Geometry Shader 所能製造的特效適合呈現爆炸或變異的場景，Proposer 想像了一個外太空的故事情節：一艘 UFO 進入場景，接著發生異常，伴隨著電流火花的效果。隨後，UFO 發生爆炸碎片四散，並以白色閃光作為過渡，緊接著外星人跳出，開始異常變異。Proposer 希望外星人在變異過程中除了長出毛髮，還會膨脹和縮小，最後分裂成多個小外星人並最終消失，營造出詭異又科幻的氛圍。

Critic 聽完後認為這是一個有創意的構想，但效果的數量稍多，可能會增加實現的難度。他建議將部分效果進行簡化，例如將電流火花效果改為燈光閃爍來表現異常，並認為小外星人分裂的實作困難度高。最終，雙方決定將爆炸效果作為結尾，應用於 UFO 與外星人上，保留故事的衝擊力，同時降低實現難度。

2) **Difficulty**

我們認為這次作業的主要難度在於了解 Geometry shader 的作用，以及它多面向的使用方法。在使用的時候我們發覺，要特別注意不同 vector 在 coordinate 的轉換是否相符，如果把 world coordinate 的 normal vector 用在 projected coordinate 的 position 上面，就會產生與原本想像完全不同的效果，甚至完全 render 不出來。經過這次作業，我們了解到藉由不同 shader 的配合，可以做出非常特別和強大的效果，真的非常有趣！

4. Work Assignment

	謝詠晴	莊婷馨
Role	Proposer	Critic
Work	● 提出大綱想法 ● CubeMap 處理 ● Alert, Fur, Noise等其他效果	● 物體的移動如自轉、公轉等 ● 最後的爆炸效果 ● 整理合併最終版程式碼

5. References

1) Alien Object File: https://free3d.com/3d-model/grayalien-v01--560376.html
2) UFO Object File: https://free3d.com/3d-model/ufo-saucer-v1--190141.html
3) Cubemap Texture:
   a. https://svs.gsfc.nasa.gov/4851#29967
   b. Convert EXR to JPG: https://www.freefileconvert.com/file/edLgblojpP_
   c. Panorama to Cubemap: https://jaxry.github.io/panorama-to-cubemap/
   d. https://www.reddit.com/r/opengl/comments/18zfenj/best_place_to_get_realistic_space_skybox_textures/?rdt=50193

6. Results

Youtube Link: https://youtu.be/KKqfotpihZM
GitHub Link: https://github.com/vch2128/2024_ICG_HW4