<進擊的苦力怕:天空墜落的復仇者>

Group 26
111550042 林筠蓁、111550082 林云歆、111550164 廖涵玉

一、Introduction

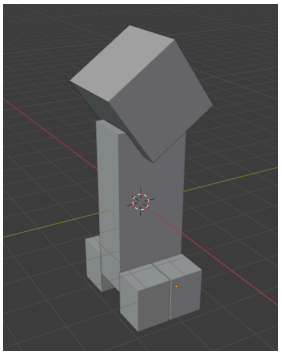
苦力怕非常害怕做苦力，於是被海綿寶寶直升機無情地丟了下去。墜落的過程中，他不幸摔斷了脖子，痛苦與屈辱交織，使他憤怒至極。他的內心怒火熊熊燃燒，最終激發了潛藏的力量。於是，苦力怕巨人化了！隨著他的力量逐漸積聚，一場震撼人心的反擊即將展開！他分別使出空間壓縮之術(抖動效果)、集氣(閃爍效果和燃燒效果)，最終以一集驚天動地的自爆(爆炸效果)完成他的復仇。

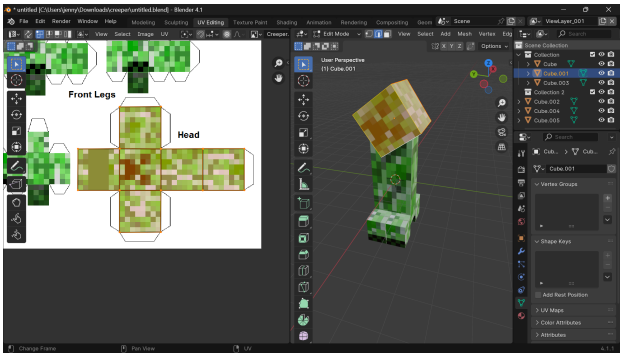
二、Implementation details

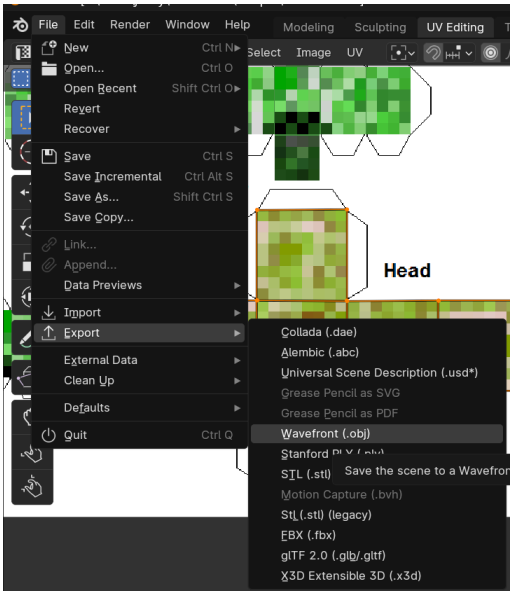
A. 製作苦力怕

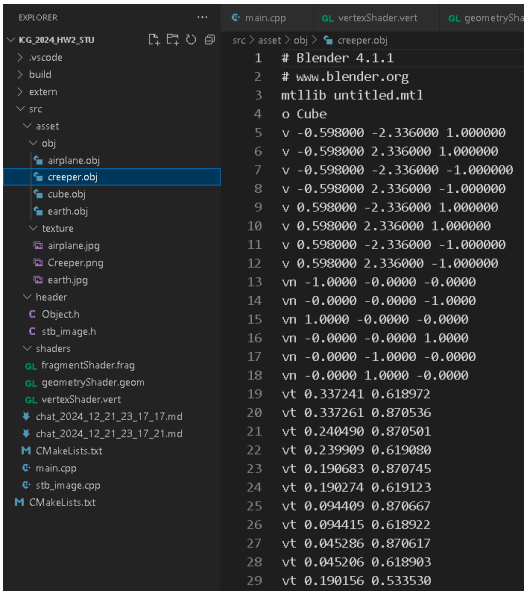
在blender畫出苦力怕模型，拿到.obj的頂點資料並匯出，拿到.obj跟texture.png之後，跟hw2一樣的步驟，就可以在openBL畫出來

1. 拉出立方體組裝他的身體形狀
2. 貼上對應身體部位的材質(一塊一塊對齊材質)
3. 匯出.obj檔









B. 巨人化(big)

按下B鍵後將isScaling轉乘true並以3的速度從1.0放大到6.5

(in main):

```
const float targetScale = 6.5f;
const float scaleSpeed = 3.0f;
if (isScaling && creeperScale < targetScale) {
    creeperScale += scaleSpeed * dt;
    if (creeperScale >= targetScale) {
        creeperScale = targetScale;
        isScaling = false;
    }
}
```

C. 抖動效果(dithering)

按下1鍵後可以轉換isDithering這個uniform參數的true和false並傳遞到geometry shader進進行抖動效果，以下是實現的細項

(geometry shader):

a. 計算三角形中心點

```
vec4 center = vec4(value: 0.0);
for (int i = 0; i < 3; i++) {
    center += gl_in[i].gl_Position;
}
center /= 3.0;
```

b. 計算從中心點指向該頂點的方向並normalize

```
for (int i = 0; i < 3; i++) {
```

```
vec4 direction = normalize(gl_in[i].gl_Position - center);
```

c. 每個頂點計算一個隨時間變化的offset產生抖動效果

```
float offsetStrength = sin(angle: time * 3.0 + float(value: i)) * 0.1;
vec4 offset4 = direction * offsetStrength;
```

d. 引入基於頂點位置和時間的noise，增加不規則抖動

```
float noiseX = sin(gl_in[i].gl_Position.x * 10.0 + time * 5.0) * 1;
float noiseY = cos(gl_in[i].gl_Position.y * 10.0 + time * 5.0) * 2;
offset4 += vec4(v0: noiseX, v1: noiseY, v2: 0.0, v3: 0.0);
```

e. 計算得到的offset應用到頂點，並設置紋理座標

```
gl_Position = gl_in[i].gl_Position + offset4;
texCoord = gs_in[i].texCoord;
```

f. 發射頂點

```
EmitVertex();
```

D. 燃燒效果(burning)

按下2鍵後轉換isBurning這個uniform參數的true和false並傳遞到fragment shader產生燃燒效果，以下為實現的細項

(in fragment shader):

a. 計算noise

```
float noise = sin(angle: texCoord.y * 10.0 + time * 3.0) * 0.5 + 0.5;
```

b. 基於y座標的顏色漸層變化

```
float gradient = texCoord.y;
```

c. 生成燃燒效果:從藍色漸變到橘色，再到紅色，最後混合黃色

```
vec3 flameColor = mix(x: vec3(v0: 0.0, v1: 0.0, v2: 1.0), y: vec3(v0: 1.0, v1: 0.5, v2: 0.0), a: gradient);
flameColor = mix(x: flameColor, y: vec3(v0: 1.0, v1: 0.0, v2: 0.0), a: noise);
flameColor = mix(x: flameColor, y: vec3(v0: 1.0, v1: 1.0, v2: 0.0), a: timeFactor);
```

d. 混合燃燒效果和原始的紋理顏色並輸出

```
vec3 finalColor = mix(x: texColor.rgb, y: flameColor, a: 0.8);
FragColor = vec4(v0: finalColor, v1: texColor.a);
```

E. 閃爍效果(flashing)

按下F鍵後轉換isFlashing這個uniform參數的true和false並傳遞到fragment shader生成苦力怕爆炸前的閃爍效果，以下為實現的細項

(in fragment shader):

a. 計算閃爍強度

```
float intensity = abs(x: sin(angle: time * 10.0));
```

b. 定義閃爍顏色為白色

```
vec3 flashColor = vec3(value: 1.0);
```

c. 混合閃爍顏色和紋理顏色

```
texColor.rgb = mix(x: texColor.rgb, y: flashColor, a: intensity);
```

F. 爆炸效果(explotion)

按下E鍵後轉換explotion這個uniform參數的true和false並傳遞到geometry shader做出爆炸效果，以下為實現的細項

(in geometry shader):

a. 取得法向量

```
vec3 normal = GetNormal();
```

```
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(v: cross(x: a,y: b));
}
```

b. 使用頂點位置、時間變量(time)透過sin和cos來生成隨機噪聲offset

```
vec3 offset3;
for (int i = 0; i < 3; i++) {
    float noiseX = sin(gl_in[i].gl_Position.x * 10.0 + time * 5.0) * 1;
    float noiseY = cos(gl_in[i].gl_Position.y * 10.0 + time * 5.0) * 2;
    offset3 = vec3(v0: noiseX, v1: noiseY, v2: 0.0);
```

c. 用explode這個函式根據位置、法向量、速度產生爆炸效果

```
gl_Position = explode(gl_in[i].gl_Position, normal + offset3, 15*i+30);
```

```
vec4 explode(vec4 position, vec3 normal, float speed){
    vec3 direction = normal * time * speed;
    return position + vec4(v0: direction, v1: 0.0);
}
```

d. 設置紋理座標並發射頂點

```
texCoord = gs_in[i].texCoord;
EmitVertex();
```

G. 消失效果(existing)
按下3鍵後按下E鍵後轉換isExist這個參數的true和false達成苦力怕出現與消失的效果

H. 飛行效果(flying)
按下4鍵後決定是否改變直升機飛行軌跡已達成不同的視覺效果

```
rotateAxisDegree += 1;
```

功能統整:(保留hw2的 A S D W H 功能)

E	F	B	1	2	3	4
explotion	flashing	big	dithering	burning	existing	flying

三、Discussion

A. Proposer:我們可以用texture來製造一隻苦力怕，製造走路接近人的效果，並閃爍後爆炸，展現像在玩minecraft一樣驚險刺激的效果
B. Critic:可是，這樣的設計聽起來有點普通，沒什麼新意。而且如果要實現追玩家的效果，苦力怕就得模擬腳的擺動，這樣一來就得把模型的每部分分開貼合 texture，很麻煩，還增加了製作的難度和時間成本。
C. Negotiator:那不如我們簡化一下流程，讓苦力怕不用移動，改成在原地旋轉，這樣可以省去模擬步態的麻煩。而且在開始閃爍之前，我們可以加一些動態特效，比如苦力

怕抖動、顏色變化、甚至是光效閃爍，讓它的爆炸感覺更加震撼。另外我們還能融合之前設計的直升機元素，讓畫面變得更豐富多元！

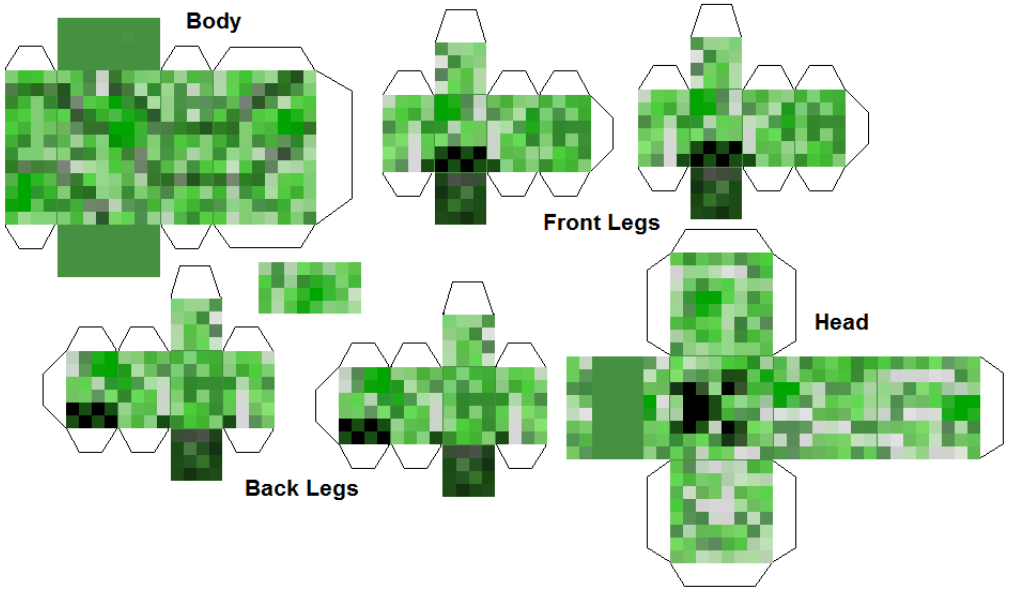
四、Work assignment

人員	工作內容
林筠蓁	Proposer，dithering，burning，影片剪輯
林云歆	Critic，Scaling，flasing
廖涵玉	Negotiator，畫苦力怕，海綿寶寶直升機

五、References

creeper texture：

https://www.minecraftforum.net/forums/mapping-and-modding-java-edition/resource-packs/1239597-req-creeper-texturing-guide



六、Results

https://youtu.be/Gy6JwiHuNJE?feature=shared