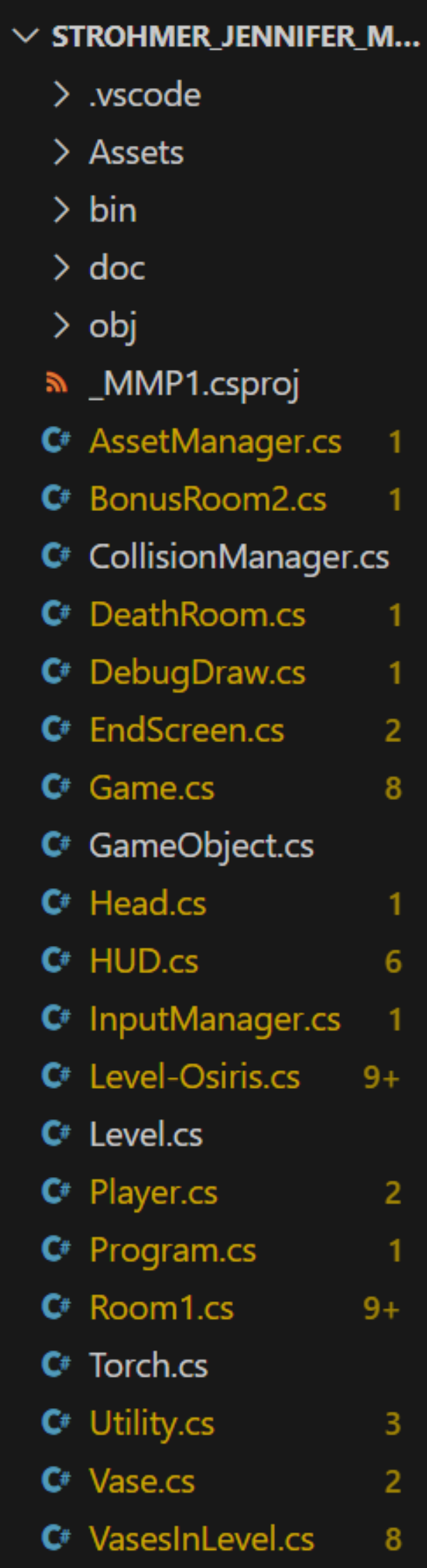


Das Projekt wurde in C# mithilfe von SFML erstellt.
Game:
In der Game-Klasse wird Level-Osiris-Klasse als Level aufgerufen.
Player.cs wird im Game.cs implementiert. Darin wird auch der Shader kontrolliert und eingefügt.
Hud, Endscreen, Clock für einen Death wurde ebenfalls eingefügt.

Level-Osiris:
In Level-Osiris.cs wird das Leveldesign erstellt. Es enthält die Logik für die Osiris-Statue.
Head.cs beinhaltet die Logik für den Kopf der Statue.
VasesInLevel.cs regelt random platzierte Vasen im Level.
DeathRoom.cs hat die Logik für einen Tod, wenn der Spieler einen falschen Raum auswählt.

Player:
In Player.cs wird die Logik vom Player kontrolliert. Hier ist das InputHandling und Animation drin, sowie Logik für Collisions.

HUD:
Das HUD wird in der HUD.cs geregelt. Es besteht aus einem torch-counter und Inventar für den Head.

EndScreen:
EndScreen.cs verwaltet das End-Menü. Der Spieler kann quitten oder neu-starten.

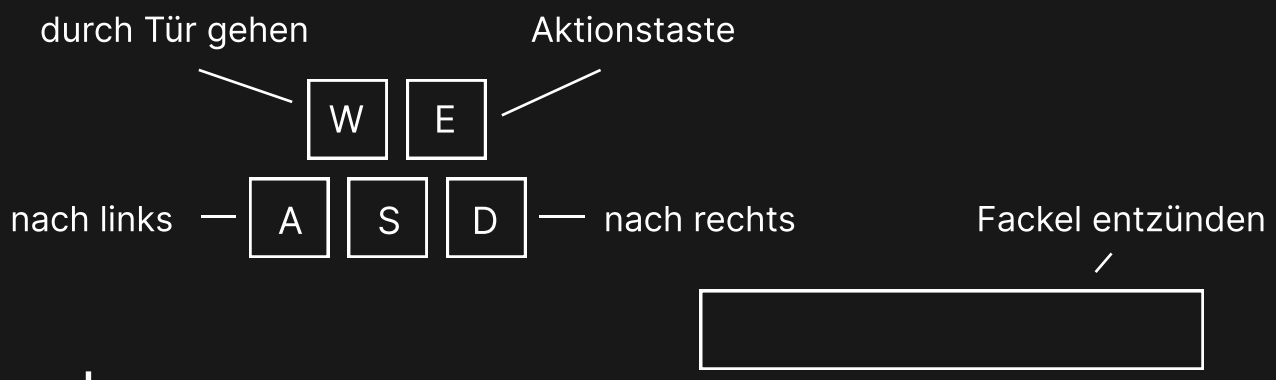
Room1:
Dies ist ein Raum mit vielen Vasen.

Vase:
Vase.cs hat die Logik eine Torch zurückzugeben, wenn es umgeworfen wurde.

Torch:
torch.cs beinhaltet die Animation
und Logik für eine Fackel

Utility.cs, Level.cs, GameObject.cs
sind Hilfsklassen

AssetManager und InputManager
sind für die Assets und Keyboard-
Input-Handling



Shader:

Assets > LightShader.frag

```
#version 120

uniform vec2 PlayerScreenPos;
uniform float coneScale;
uniform float time;
const float flickerThreshold = 0.1; // Adjust the threshold value as needed
const float flickerRange = 0.9; // Adjust the range value as needed

void main()
{
    // Calculate the distance from the player position to the current fragment position
    float dist = distance(gl_TexCoord[0].xy, PlayerScreenPos);

    if (dist > coneScale)
    {
        gl_FragColor = vec4(0.0, 0.0, 0.0, 1.0);
    }
    else
    {
        // Calculate the distance from the fragment position to the center of the cone
        float centerDist = abs(dist - coneScale);

        // Calculate the flicker amount based on the distance from the cone edge to halfway towards the center
        float flickerAmount = abs(cos(time * 10.0) * sin(time * 12.0) * (1.0 - centerDist / coneScale));

        // Apply the threshold and range
        flickerAmount = max(flickerAmount - flickerThreshold, 0.0) / flickerRange;

        // Soften the flicker effect
        flickerAmount = smoothstep(0.1, 0.9, flickerAmount);

        gl_FragColor = vec4(0.0, 0.0, 0.0, flickerAmount);
    }
}

```