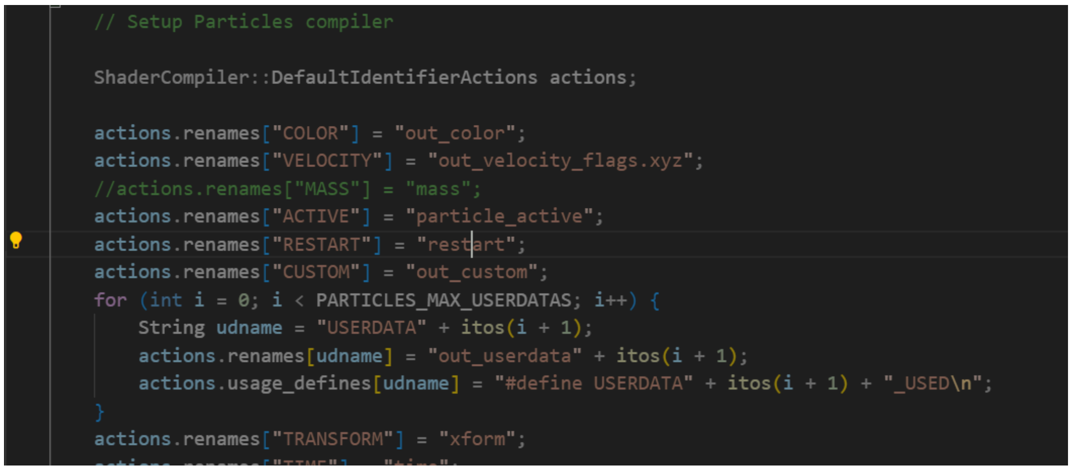
CHANGELOG:

<u>Within:</u>

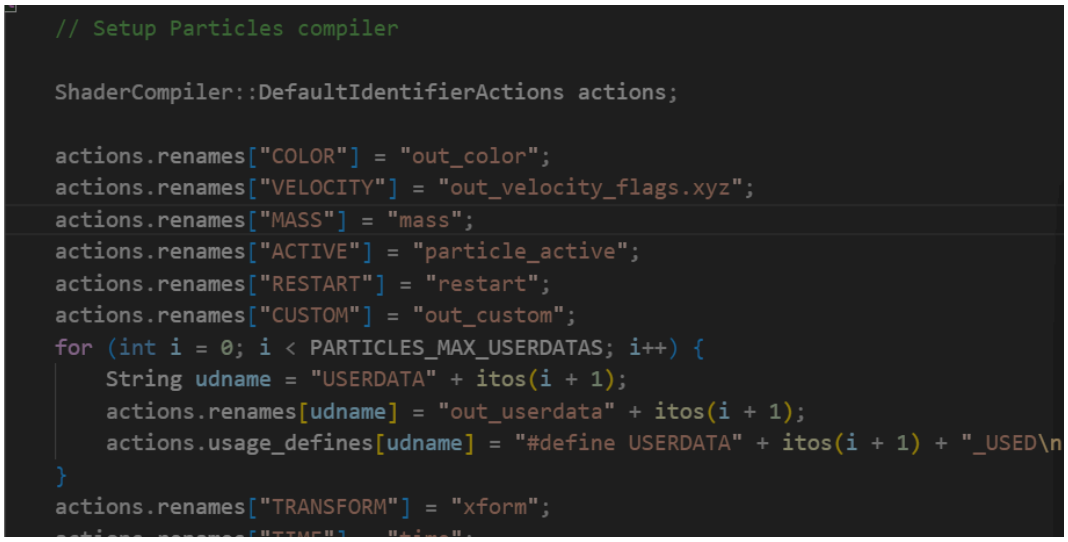
drivers/gles3/storage/material_storage.cpp

Before



```
// Setup Particles compiler

ShaderCompiler::DefaultIdentifierActions actions;

actions.renames["COLOR"] = "out_color";
actions.renames["VELOCITY"] = "out_velocity_flags.xyz";
//actions.renames["MASS"] = "mass";
actions.renames["ACTIVE"] = "particle_active";
actions.renames["RESTART"] = "restart";
actions.renames["CUSTOM"] = "out_custom";
for (int i = 0; i < PARTICLES_MAX_USERDATAS; i++) {
    String udname = "USERDATA" + itos(i + 1);
    actions.renames[udname] = "out_userdata" + itos(i + 1);
    actions.usage_defines[udname] = "#define USERDATA" + itos(i + 1) + "_USED\n";
}
actions.renames["TRANSFORM"] = "xform";
```

After

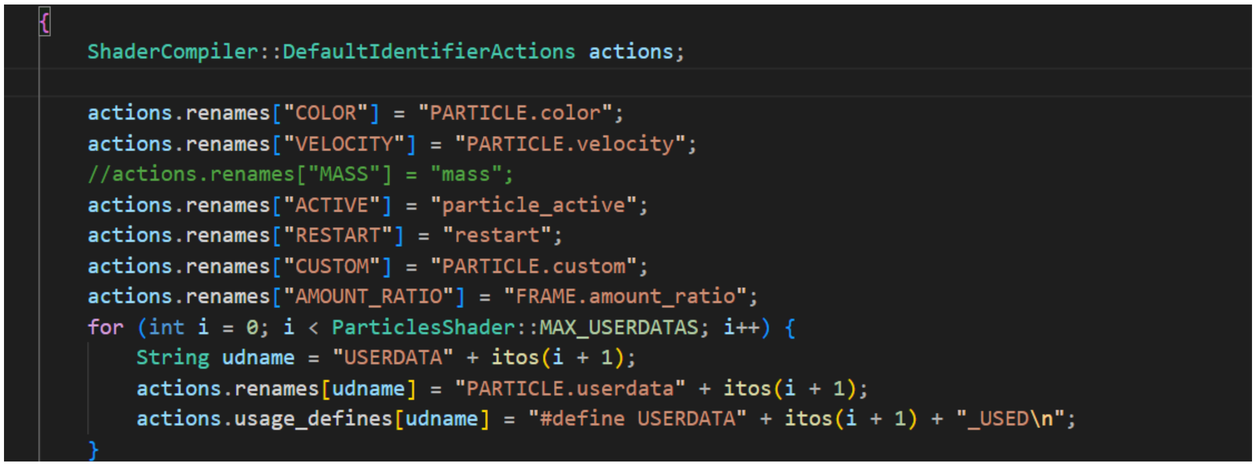


```
// Setup Particles compiler

ShaderCompiler::DefaultIdentifierActions actions;

actions.renames["COLOR"] = "out_color";
actions.renames["VELOCITY"] = "out_velocity_flags.xyz";
actions.renames["MASS"] = "mass";
actions.renames["ACTIVE"] = "particle_active";
actions.renames["RESTART"] = "restart";
actions.renames["CUSTOM"] = "out_custom";
for (int i = 0; i < PARTICLES_MAX_USERDATAS; i++) {
    String udname = "USERDATA" + itos(i + 1);
    actions.renames[udname] = "out_userdata" + itos(i + 1);
    actions.usage_defines[udname] = "#define USERDATA" + itos(i + 1) + "_USED\n
}
actions.renames["TRANSFORM"] = "xform";
```

<u>Within:</u>

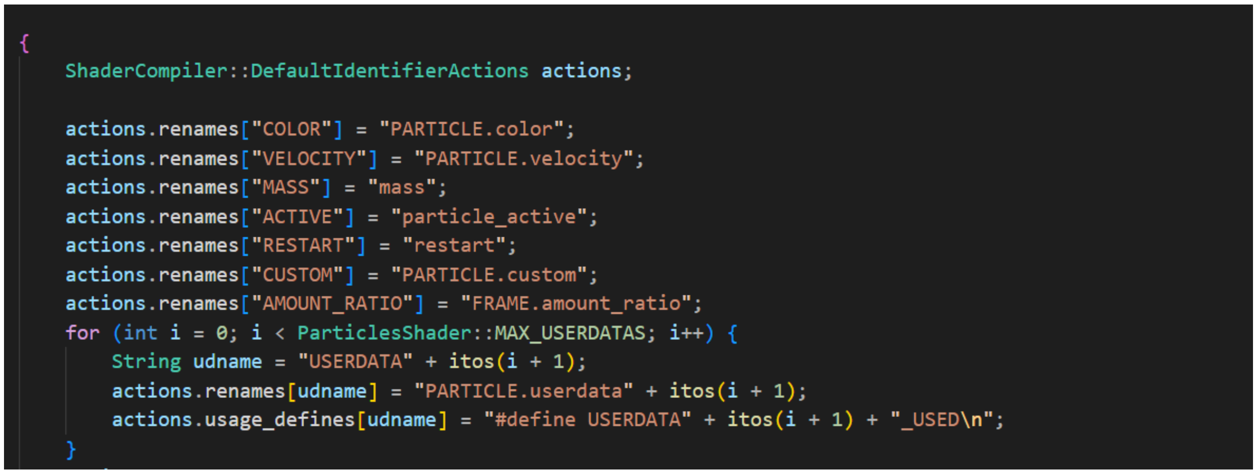
servers/rendering/renderer_rd/storage_rd/particles_storage.cpp:

Before



```
{
    ShaderCompiler::DefaultIdentifierActions actions;

    actions.renames["COLOR"] = "PARTICLE.color";
    actions.renames["VELOCITY"] = "PARTICLE.velocity";
    //actions.renames["MASS"] = "mass";
    actions.renames["ACTIVE"] = "particle_active";
    actions.renames["RESTART"] = "restart";
    actions.renames["CUSTOM"] = "PARTICLE.custom";
    actions.renames["AMOUNT_RATIO"] = "FRAME.amount_ratio";
    for (int i = 0; i < ParticlesShader::MAX_USERDATAS; i++) {
        String udname = "USERDATA" + itos(i + 1);
        actions.renames[udname] = "PARTICLE.userdata" + itos(i + 1);
        actions.usage_defines[udname] = "#define USERDATA" + itos(i + 1) + "_USED\n";
    }
```

After

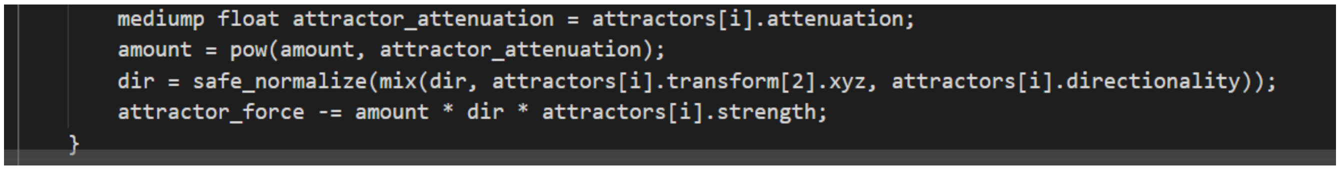


```
{
    ShaderCompiler::DefaultIdentifierActions actions;

    actions.renames["COLOR"] = "PARTICLE.color";
    actions.renames["VELOCITY"] = "PARTICLE.velocity";
    actions.renames["MASS"] = "mass";
    actions.renames["ACTIVE"] = "particle_active";
    actions.renames["RESTART"] = "restart";
    actions.renames["CUSTOM"] = "PARTICLE.custom";
    actions.renames["AMOUNT_RATIO"] = "FRAME.amount_ratio";
    for (int i = 0; i < ParticlesShader::MAX_USERDATAS; i++) {
        String udname = "USERDATA" + itos(i + 1);
        actions.renames[udname] = "PARTICLE.userdata" + itos(i + 1);
        actions.usage_defines[udname] = "#define USERDATA" + itos(i + 1) + "_USED\n";
    }
```

Within:

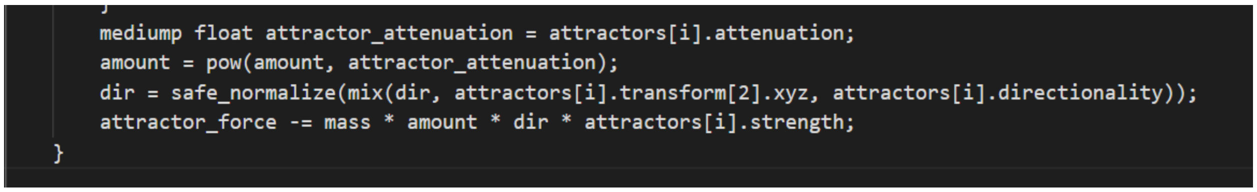
drivers/gles3/shaders/particles.glsl

Before



```
        mediump float attractor_attenuation = attractors[i].attenuation;
        amount = pow(amount, attractor_attenuation);
        dir = safe_normalize(mix(dir, attractors[i].transform[2].xyz, attractors[i].directionality));
        attractor_force -= amount * dir * attractors[i].strength;
    }
```

After

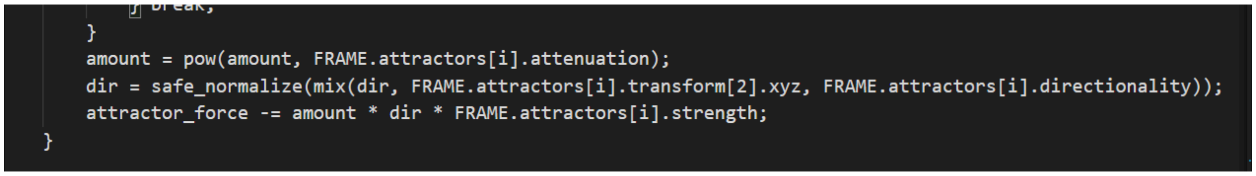


```
        }
        mediump float attractor_attenuation = attractors[i].attenuation;
        amount = pow(amount, attractor_attenuation);
        dir = safe_normalize(mix(dir, attractors[i].transform[2].xyz, attractors[i].directionality));
        attractor_force -= mass * amount * dir * attractors[i].strength;
    }
```

<u>Within:</u>

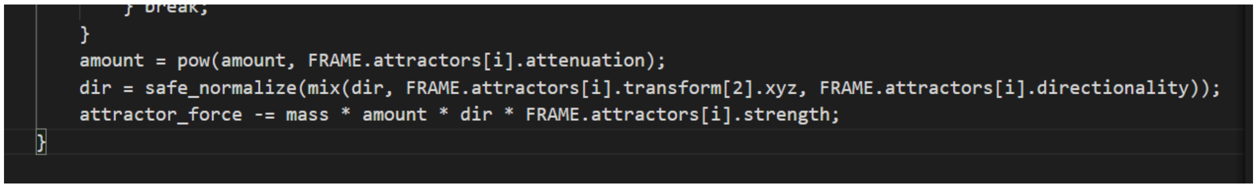
servers/rendering/renderer_rd/shaders/particles.glsl

Before



```
            } break;
        }
        amount = pow(amount, FRAME.attractors[i].attenuation);
        dir = safe_normalize(mix(dir, FRAME.attractors[i].transform[2].xyz, FRAME.attractors[i].directionality));
        attractor_force -= amount * dir * FRAME.attractors[i].strength;
    }
```

After



```
            } break;
        }
        amount = pow(amount, FRAME.attractors[i].attenuation);
        dir = safe_normalize(mix(dir, FRAME.attractors[i].transform[2].xyz, FRAME.attractors[i].directionality));
        attractor_force -= mass * amount * dir * FRAME.attractors[i].strength;
    }
```