Assignment 4 Report

By Fedor Ivachev 费杰 2019280373

Screenshot of the program

Simple particle effect: Snowing

1) Particle effect of snowing

In this program a particle effect of snowing is achieved by modifying tutorial code to follow the requirements.

Particles are stored in the vector of ‘SnowFlake’ objects. Each of the SnowFlake objects is defined by private values:

glm::vec3 gravAcceleratio; - acceleration of gravity

glm::vec3 initialSpeed; - initial speed of snowflake

double initialLife; - initial lifetime of snowflake

glm::vec3 force; - force applied to the snowflake

glm::vec3 speed; - current speed of the snowflake

double mass; - mass of the snowflake

double life; - current life of the snowflake

And the public values:

double scale = 10.0f; - size of the snowlake

glm::vec3 position; - current position on the screen

glm::vec4 color; - color of the snowflake

By processing each of the particles, the new values are counted every moment and each of the particles is rendered. If lifetime is close to zero,

speed = initialSpeed;

color = glm::vec4(1.0f, 1.0f, 1.0f, 0.5f);

position = newPos;

life = initialLife;

Repeating it every frame, the particle effect is achieved

2) Snowflakes have different sizes.

The scale of each particle is counted randomly during its initialization: 10.0f + rand() % 40

3) Start with less snowflakes and gradually increase the number over time.

Every frame a new particle can be added to the array with a probability of 1/1000.

```
if (rand() % (1000) == 0) {
        snowflakeNum++;
        snowflakes.push_back(SnowFlake(0.5, glm::vec3(rand() % (WIDTH), HEIGHT + 10, 0),
           glm::vec3(0.0, rand() % 7 * -1, 0.0), glm::vec3(0.0, -0.5, 0.0), (double)rand() /
(double)RAND_MAX * 10 + 32, 10.0f + rand() % 40));
```

The background is rendered as a texture bind to the quad.

These are the vertex and fragment shaders:

main.cpp | ParticleSystem.h | main.vert.glsl | main.frag.glsl

```
#version 330 core
// <vec2 position, vec2 texCoords>
layout (location = 0) in vec4 vertex;

out vec2 TexCoords;
out vec4 ParticleColor;

uniform mat4 projection;
uniform vec2 offset;
uniform vec4 color;
uniform float scale;

void main(){
    TexCoords = vertex.zw;
    ParticleColor = color;
    gl_Position = projection * vec4((vertex.xy * scale) + offset, 0.0, 1.0);
}
```

main.cpp | ParticleSystem.h | main.vert.glsl | main.frag.glsl

```
#version 330 core
in vec2 TexCoords;
in vec4 ParticleColor;
out vec4 color;

uniform sampler2D sprite;

void main(){
    color = (texture(sprite, 1 - TexCoords) * ParticleColor);
}
```

The video demonstration of the app is captured by Windows 10 built-in Xbox game bar APP.