Particle

1. Why did you implement particle?
    A. The reason why I implement particle is I don't know about particle system. When I ask to professor what is your recommended one new features for B grade, he told me that he expect animation or particle system. I was able to determine to implement animation. Animation is easy for me because I already implemented animation in my gam 200 engine. Although I do not know high level of animation kind of spine, I can make animation seems work. However, I think it is silly and the behavior that make me stupid. Thus, I choose to implement particle system to learn about new graphics stuff.
2. What is particle?
    A. Simply, Particle is a set of tiny sprites. They are managed and emitted by particle emitter or particle generator. Each rectangles of particle have different life span. If one of them is dead, it become hibernation particle and wait to respawn.
3. How did you implement it?
    A. First of all, I implement particle struct.

```
// Declare particle struct
struct Particle
{
    vector2<float> position, velocity;
    Graphics::Color4f color;
    float life;

    Particle()
        : position(repeatedFloat: 0.f), velocity(repeatedFloat: 0.f), color(grey: 1.f), life(0.f) {}
};
```

    B. Currently, it has position, velocity, color value, and life span.

```
// Particle member variable
const static int particleSize = 500;
Particle particles[particleSize];
```

    C. At the same time, I add an array of Particle.
    D. Currently the size of it is 500. It is a given number from the particle tutorial website that I referenced. It can be changed arbitrarily.

```
// Sketch::Update();
const unsigned numOfNewParticlePerFrame = 1;

// Add new partices
for (unsigned int i = 0; i < numOfNewParticlePerFrame; ++i)
{
    const int unusedParticle = FirstUnusedParticle(particles);
    const float xRandom = (rand() % 500) - 250;
    const float  yRandom = rand() % 5000;
    object.velocity = vector2<float>{ xRandom, yRandom};
    vector2<float> offset{ repeatedFloat: -100.f};
    RespawnParticle(particles[unusedParticle], [&]object, offset);
}
```

E. In my version, there is no explicitly different shader with texture shader. It can be changed also arbitrarily and I guess it is one way to make particle fancy.

F. What should do in at the beginning of each frame is add new particle.

G. The variable numOfNewParticlePerFrame indicated the number of particle that invoked in each frame.

```
void Sketch::RespawnParticle(Particle& particle, Object& object, vector2<float> offset)
{
    float rColor = ((rand() % 100) / 100.f);
    float gColor = ((rand() % 100) / 100.f);
    float bColor = ((rand() % 100) / 100.f);

    particle.position.x = object.position.x + offset.x;
    particle.position.y = object.position.y + offset.y;

    particle.color = Graphics::Color4f{ red: rColor, green: gColor, blue: bColor, alpha: 1.f };
    particle.life = 5.f;
    particle.velocity = object.velocity * 0.1f;
}
```

H. Loop try to find the index of unused particle from particle array.

I. After find one particle, get a velocity with a random value via the function rand() and initialize it again.

```
// Update all particles
for (unsigned int i = 0; i < particleSize; ++i)
{
    Particle& p = particles[i];
    p.life -= dt; // reduce life
    if (p.life > 0.f)
    {  // particle is alive, thus update
        p.position += p.velocity * dt;
        p.color.alpha -= dt * 0.5f;
        if (p.color.alpha <= 0.f)
        {
            p.color.alpha = 0.f;
        }
    }
}
```

J. After respawn it, update every particle.

K. First, remove their life. If it is still alive, update their value.

L. Finally, render it!

M. That's the way I've implemented particle.

N. Reference: https://learnopengl.com/In-Practice/2D-Game/Particles

4. How did I make another particle

A. Poison pot

i. Everything is based on example particle.

ii. I changed on velocity, position, color when instantiate particle.

```
const float xRandomPosition = position.x + static_cast<float>((rand() % 300) - 150);
const float yPosition = position.y;
const float xRandomVelocity = static_cast<float>(rand() % 1000) - 500;
const float yRandomVelocity = static_cast<float>(rand() % 500);
obj.position = vector2{ xRandomPosition, yPosition };
obj.velocity = vector2{ xRandomVelocity, yRandomVelocity };
RespwanParticle(smokeParticle[unusedParticle], [&]obj, vector2{ repeatedFloat:0.f }, Graphics::Color4f{ red:0.545f, green:0.f, blue:0.545f });
```

iii. In update,

```
p.position += p.velocity * dt;
p.velocity.y -= 1.f;
```

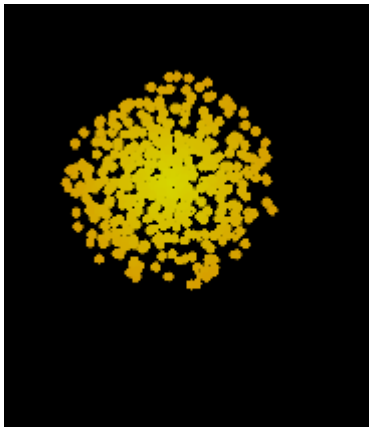
```
p.color.alpha -= dt * 0.25f;
if (p.color.alpha <= 0.f)
{
    p.color.alpha = 0.f;
}
```

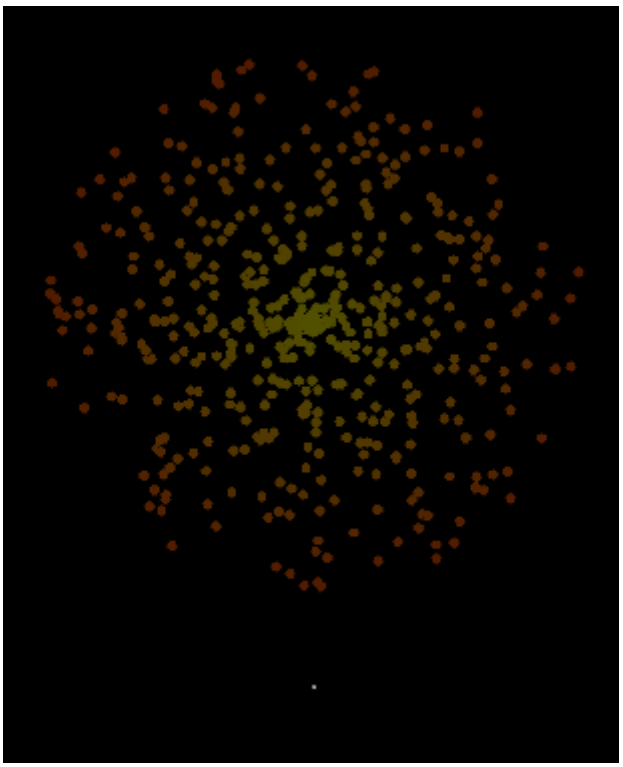
iv. Position update with their velocity, velocity decrease its y-axis velocity because of gravity. In addition its color is going to become transparent.

v. The thing that I believe cool in this particle is …

```
if (p.position.y <= position.y - 150.f)
{
    p.velocity.y = (-p.velocity.y) / 2.f;
}
```

vi. I made law of action and reaction. Thus, when particles come to specific y-axis position, their y-axis velocity flipped. As a result they go to upward.





B. Firework explosion particle

i. There are a bunch of difference with basic example particle.

ii. First of all, start position and velocity is different.

```
const float xRandomPosition = position.x;
const float yPosition = position.y;
const float radius = static_cast<float>(rand() % 1000);
const float radian = static_cast<float>(rand() % 314000) * 0.0001f;

const float xRandomVelocity = radius * cos(radian);
float yRandomVelocity = radius * sin(radian);

// Half of their y-axis velocity is flipped
if (i % 2)
{
    yRandomVelocity = -yRandomVelocity;
}
```

iii. Like this, their position is fixed with given position and their velocity will get a circular randomly. Thus they get a radius and radian value with random and calculate to get an x and y position with trigonometric function.

```
static constexpr float MAXIMUM_COLOR_VALUE = 1.f;
static constexpr float RATIO = 200.f;
const float t = magnitude(vector p.position - position);
p.color.green = (MAXIMUM_COLOR_VALUE - (t / RATIO));

p.color.alpha -= dt * 0.25f;
if (p.color.alpha <= 0.f)
{
    p.color.alpha = 0.f;
}
```

iv. Furthermore, I've implement this to represent that color going to become red when they are far away from the start position of explosion.