COMP710: Studio Session 07 – Exercise:

EXERCISE NAME: *C++ – Particle Effects*

Create **Particle** and **ParticleEmitter** classes. Generate basic particle effects and implement a Dear ImGui debug interface for experimenting with and tweaking/tuning the particle emitter.

For example, the **particle.h** interface:

```
// COMP710 GP Framework
#ifndef __PARTICLE_H__
#define __PARTICLE_H__

// Local includes:
#include "vector2.h"

// Forward declarations:
class Renderer;
class Sprite;

// Class declaration:
class Particle
{
    // Member methods:
public:
    Particle();
    ~Particle();

    bool Initialise(Sprite& sprite);
    void Process(float deltaTime);
    void Draw(Renderer& renderer);

protected:

private:
    Particle(const Particle& particle);
    Particle& operator=(const Particle& particle);

    // Member data:
public:
    Sprite* m_pSharedSprite;
    Vector2 m_postion;
    Vector2 m_velocity;
    Vector2 m_acceleration;
    float m_fMaxLifespan;
    float m_fCurrentAge;
    float m_fColour[3];
    bool m_bAlive;

protected:

private:

};

#endif // __PARTICLE_H__
```

And the associated **`particle.cpp`** implementation:

```
// COMP710 GP Framework

// This include:
#include "particle.h"

// Local includes:
#include "renderer.h"
#include "sprite.h"

Particle::Particle()
: m_bAlive(false)
, m_fCurrentAge(0.0f)
{

}

Particle::~Particle()
{

}

bool
Particle::Initialise(Sprite& sprite)
{
    m_pSharedSprite = &sprite;
    return true;
}

void
Particle::Process(float deltaTime)
{
    if (m_bAlive)
    {
        m_fCurrentAge += deltaTime;
        m_velocity += m_acceleration * deltaTime;
        m_postion += m_velocity * deltaTime;

        if (m_fCurrentAge > m_fMaxLifespan)
        {
            m_bAlive = false;
        }
    }
}

void
Particle::Draw(Renderer& renderer)
{
    if (m_bAlive)
    {
        m_pSharedSprite->SetRedTint(m_fColour[0]);
        m_pSharedSprite->SetGreenTint(m_fColour[1]);
        m_pSharedSprite->SetBlueTint(m_fColour[2]);
        float alpha = 1.0f - (m_fCurrentAge / m_fMaxLifespan);
        m_pSharedSprite->SetAlpha(alpha);
        m_pSharedSprite->SetX(m_postion.x);
        m_pSharedSprite->SetY(m_postion.y);
        m_pSharedSprite->Draw(renderer);
    }
}
```

And an example of the **particleemitter.h** interface:

```
// COMP710 GP Framework
#ifndef __PARTICLEEMITTER_H__
#define __PARTICLEEMITTER_H__

// Library includes:
#include <vector>

// Forward declarations:
class Renderer;
class Sprite;
class Particle;

// Class declaration:
class ParticleEmitter
{
    // Member methods:
public:
    ParticleEmitter();
    ~ParticleEmitter();

    bool Initialise(Renderer& renderer);
    void Process(float deltaTime);
    void Draw(Renderer& renderer);

    void Spawn();

    void DebugDraw();

protected:

private:
    ParticleEmitter(const ParticleEmitter& particleemitter);
    ParticleEmitter& operator=(const ParticleEmitter& particleemitter);

    // Member data:
public:

protected:
    Sprite* m_pSharedSprite;
    std::vector<Particle*> m_particles;

    float m_fTimeElapsed;

    int m_iSpawnBatchSize;
    float m_fEmitRate;
    float m_fMaxLifespan;
    float m_fAccelerationScalar;
    float m_fColour[3];
    float m_fMinAngle;
    float m_fMaxAngle;
    float m_fX;
    float m_fY;

private:

};

#endif // __PARTICLEEMITTER_H__
```

The **`ParticleEmitter`** implementation:

- **`Initialise`**:
    - Creates the shared sprite to use for all particles emitted by this emitter.
- **`Process`**:
    - Processes all particles within its member container.
    - Emits new particles, adding them to its member container.
    - Removes dead particles form its container.
- **`Draw`**:
    - Draws all particles in its member container.
- **`DebugDraw`**:
    - Dear ImGui interface for tuning/tweaking particle emitter properties.
- **`Spawn`**:
    - Creates a new particle.
    - Sets up initial state of new particle.
        - Alive…
        - Max lifespan…
        - Starting position…
        - Starting acceleration…
        - Starting colour…
    - Adds the new particle to the emitter's member container