```cpp
#include "stdafx.h"
//--managers
#include "GameObjectManager.h"
#include "GameObjectFactory.h"
#include "Global/EnemyType.h"
#include "Global/PowerUpType.h"
#include "Global/ParticleType.h"
//--components
#include "Component/BoxCollider.h"
#include "Component/Health.h"
#include "Component/SpriteRenderer.h"
#include "Component/MovementInput.h"
#include "Component/PlayerShooter.h"
#include "Component/BulletMovement.h"
#include "Component/Circle.h"
#include "Component/EnemyMovement.h"
#include "Component/EnemyRandomMovement.h"
#include "Component/EnemySpawner.h"
#include "Component/EnemyShooter.h"
#include "Component/HitEffect.h"
#include "Component/EnemySplit.h"
#include "Component/EnemyDefense.h"
#include "Component/Particle.h"

namespace GameObjectFactory
{
    Object::Ref CreatePlayer(const Vector2& center, float distance, float scale)
    {
        Object::Ref player = GameObjectManager::GetInstance().AddToManager();
        player->SetScale(scale);

        //---components
        //sprite
        auto& player_sprite = player->AddComponent<SpriteRenderer>(".\\Data\\Sprite\\player.png", 4, 1);
        //player_sprite.CreateAnimation(0, 0.05f, { 0,1,2,3 });
        //player_sprite.SetAnimation(0);
        //collider
        Vector2 collider_size{ player_sprite.width() * 0.9f, player_sprite.height() * 0.6f };
        auto& player_collider = player->AddComponent<BoxCollider>("player", collider_size);
        //movement input, health & shooting input
        player->AddComponent<MovementInput>(center, distance);
        player->AddComponent<Health>(50);
        player->AddComponent<PlayerShooter>(25.f);
        player->AddComponent<HitEffect>(0.1f);
```

```
45
46          return player;
47      }
48
49      Object::Ref CreatePowerUp(PowerUpType powerup_type)
50      {
51          Object::Ref power_up = GameObjectManager::GetInstance
              ().AddToManager();
52
53          switch (powerup_type)
54          {
55          case PowerUpType::Health:
56              //sprite
57              auto& sprite = power_up->AddComponent<SpriteRenderer>(".\\Data
                  \\Sprite\\PowerUp_05.png", 1, 1);
58              sprite.SetScale(.18f);
59              //collider
60              Vector2 collider_size{ sprite.width(), sprite.height() };
61              power_up->AddComponent<BoxCollider>("health_power",
                  collider_size);
62              break;
63          }
64
65          return power_up;
66      }
67
68      Object::Ref CreateEnemy(Transform& transform, EnemyType enemy_type)
69      {
70          Object::Ref enemy = GameObjectManager::GetInstance().AddToManager
              (transform);
71          SpriteRenderer& enemy_sprite = enemy->AddComponent<SpriteRenderer>
              (".\\Data\\Sprite\\enemies.png", 4, 5);
72          //collider
73          Vector2 collider_size{ enemy_sprite.width(), enemy_sprite.height
              () / 2 };
74          auto& enemy_collision = enemy->AddComponent<BoxCollider>("enemy",
              collider_size);
75          //hit effect, input the scale offset
76          enemy->AddComponent<HitEffect>(0.2f);
77
78          switch (enemy_type)
79          {
80          case EnemyType::SlowChaseType:
81              //health
82              enemy->AddComponent<Health>(200);
83              //movement
84              enemy->AddComponent<EnemyMovement>(0.04f);
85              //sprite
86              enemy_sprite.SetFrame(3);
```

```
87              break;
88          case EnemyType::ShootType:
89              enemy->AddComponent<Health>(100);
90              //shooter
91              enemy->AddComponent<EnemyShooter>(20.f);
92              //movement
93              enemy->AddComponent<EnemyMovement>(0.2f);
94              //sprite
95              enemy_sprite.SetFrame(19);
96              break;
97          case EnemyType::TwoModeType:
98              //health
99              enemy->AddComponent<Health>(120);
100             //movement
101             enemy->AddComponent<EnemyMovement>(0.05f);
102             //sprite
103             enemy_sprite.SetFrame(15);
104             enemy_sprite.SetColor(0.f, 1.f, 0.212f);
105             //shooter
106             enemy->AddComponent<EnemyShooter>(20.f);
107             enemy->AddComponent<EnemyDefense>();
108             break;
109         case EnemyType::SplitType:
110             //health
111             enemy->AddComponent<Health>(60);
112             //movement
113             enemy->AddComponent<EnemyMovement>(0.03f);
114             enemy->AddComponent<EnemyRandomMovement>();
115             //sprite
116             enemy_sprite.SetFrame(11);
117             //split
118             enemy->AddComponent<EnemySplit>(2);
119             break;
120         case EnemyType::FastChaseType:
121             //health
122             enemy->AddComponent<Health>(60);
123             //movement
124             enemy->AddComponent<EnemyMovement>(0.1f);
125             //sprite
126             enemy_sprite.SetFrame(7);
127             break;
128         }
129         return enemy;
130     }
131
132     Object::Ref CreateBullet(float r, float g, float b, float size, int
        lifespan, float speed, std::string tag)
133     {
134         Object::Ref bullet = GameObjectManager::GetInstance().AddToManager
```

```
            ();
135         bullet->GetComponent<Transform>().scale = size;
136         //sprite
137         auto& bullet_sprite = bullet->AddComponent<SpriteRenderer>(".\
              \Data\\Sprite\\bullet.png", 1, 1);
138         bullet_sprite.SetColor(r, g, b);
139         //collider
140         Vector2 collider_size{ bullet_sprite.width(), bullet_sprite.height
              () };
141         bullet->AddComponent<BoxCollider>(tag, collider_size);
142         //bullet auto movement
143         bullet->AddComponent<BulletMovement>(lifespan, speed);
144
145         return bullet;
146     }
147
148     Object::Ref CreateCombatPlanet(const Vector2& center_position, float
          radius, int steps)
149     {
150         Transform circle_transfrom{ center_position, radius };
151         Object::Ref circle = GameObjectManager::GetInstance().AddToManager
              (circle_transfrom);
152         //shape
153         circle->AddComponent<Circle>(center_position, radius, steps);
154
155         return circle;
156     }
157
158     Object::Ref CreateEnemySpawner(const Vector2& spawn_pos)
159     {
160         Transform transform{ spawn_pos, 0.f };
161         Object::Ref spawner = GameObjectManager::GetInstance
              ().AddToManager(transform);
162         //enemy spawner
163         spawner->AddComponent<EnemySpawner>();
164
165         return spawner;
166     }
167
168     Object::Ref CreateParticle(float r, float g, float b, float size,
          float lifespan, ParticleType type)
169     {
170         Object::Ref particle = GameObjectManager::GetInstance
              ().AddToManager();
171         particle->GetComponent<Transform>().scale = size;
172
173         switch (type)
174         {
175         case ParticleType::Explosion:
```

```
176         {
177             //sprite
178             auto& particle_sprite = particle->AddComponent<SpriteRenderer> ↵
                  (".\\Data\\Sprite\\flower.png", 1, 1);
179             particle_sprite.SetColor(r, g, b);
180             break;
181         }
182         case ParticleType::Healing:
183         {
184             auto& particle_sprite = particle->AddComponent<SpriteRenderer> ↵
                  (".\\Data\\Sprite\\health.png", 1, 1);
185             particle_sprite.SetColor(r, g, b);
186             break;
187         }
188         }
189         particle->AddComponent<Particle>(lifespan);
190 
191         return particle;
192     }
193 }
194 
```