I. Wstęp Teoretyczny

Systemy cząsteczek służą do modelowania zjawisk stochastycznych i amorficznych poprzez zarządzanie zbiorem prymitywów graficznych, z których każdy definiowany jest przez wektor stanów (położenie, prędkość, kolor, czas życia). W środowisku Unity system ten oparty jest na architekturze modułowej, reprezentowanej w API przez struktury danych (types). Ze względu na fakt, iż moduły takie jak MainModule czy EmissionModule są typami wartościowymi, ich modyfikacja programowa wymaga pobrania struktury do zmiennej lokalnej, edycji parametrów i ponownego przypisania jej do komponentu ParticleSystem.

Zaawansowana kontrola nad symulacją realizowana jest poprzez bezpośrednią manipulację buforem cząsteczek. Proces ten obejmuje ekstrakcję danych o aktywnych jednostkach do tablicy w przestrzeni CPU za pomocą metody GetParticles, implementację autorskich algorytmów transformacji wewnątrz pętli iteracyjnej, a następnie przesłanie zaktualizowanego zbioru do silnika poprzez SetParticles. Interakcja z otoczeniem odbywa się natomiast poprzez system zdarzeń (np. OnParticleTrigger), który pozwala na selektywną zmianę stanów cząsteczek wchodzących w kolizję z obiektami sceny. Monitorowanie cyklu życia systemu za pomocą metody IsAlive stanowi kluczowy element optymalizacji zasobów obliczeniowych.

II. Zadania do wykonania

Zadanie 1

Stwórz skrypt, który pobierze komponent ParticleSystem i przejmie kontrolę nad jego modułem głównym. Zaimplementuj logikę, która pozwala za pomocą klawiatury lub myszy dynamicznie zmieniać siłę grawitacji oraz czas życia nowo powstających cząsteczek.

Zadanie 2

Zmodyfikuj system tak, aby przestał generować cząsteczki automatycznie (wyłącz emisję w edytorze). Rozbuduj skrypt o funkcję, która w odpowiedzi na konkretne zdarzenie (np. kliknięcie lewym przyciskiem myszy) wygeneruje gwałtowny wystrzał (Burst) określonej liczby cząsteczek.

Zadanie 3

Wykorzystaj tablicę cząsteczek (GetParticles), aby tchnąć życie w obiekty wystrzelone w poprzednim zadaniu. Napisz pętlę, która dla każdej żyjącej cząsteczki obliczy wektor kierunkowy w stronę wyznaczonego celu (innego obiektu na scenie). Cząsteczki powinny korygować swój tor lotu w każdej klatce, dążąc do punktu docelowego.

Zadanie 4

Skonfiguruj system tak, aby reagował na kontakt z obiektami otoczenia. Wykorzystaj metodę OnParticleTrigger(), aby wykryć moment, w którym poszczególne cząsteczki uderzają w cel. Zamiast znikać, cząsteczki po kontakcie powinny zmienić swój stan (np. gwałtowna zmiana koloru lub wielkości), informując gracza o trafieniu.

Zadanie 5

Dokończ skrypt, implementując mechanizm monitorujący stan systemu. Jeśli system przestał emitować nowe cząsteczki, a wszystkie istniejące zakończyły swój żywot (dotarły do celu lub wygasły), obiekt powinien zostać automatycznie zniszczony lub dezaktywowany, aby zwolnić zasoby procesora.