Programowanie komputerów 2 - Teleinformatyka
Laboratorium 2

Implementacje wszystkich metoda mogą być implementowane w plikach `*.h` dla oszczędności czasu.
Zapoznać się z zawartością plików: `ProjectileCollection.h`, `Simulation.h`

1. (**1 pkt**) W klasie `BattlefieldSimulation` zaimplementować wszystkie metody czysto wirtualne klasy bazowej `Simulation`. Klasa ma dostęp do wszystkich obiektów klasy `Projectile` (pole chronione `projectiles` klasy `Simulation`). W implementowanej metodzie `Init` odwołaj się do tego obiektu i dodaj nowy pocisk (obiekt klasy `Projectile`). Po tym kod powinien się kompilować, uruchom i sprawdź działanie programu.
2. (**1 pkt**) Utwórz klasę `Vector2D` reprezentującą wektor w przestrzeni 2 wymiarowej. Klasa powinna zawierać konstruktor inicjujący współrzędne. Utwórz dla tej klasy przeciążone operatory:
   (a) przypisania,
   (b) predekrementacji i postdekrementacji dekrementujace `x` i `y`,
   (c) `operator[]` dla indeksu 0 zwraca `x`, dla innego `y`
   (d) `operator+=`,
   (e) `operator*` przemnażający współrzędne wektora przez skalar,
   (f) metodę `normalize` normalizującą wektor (wektor znormalizowany ma długość 1),
   (g) metodę `Show` wyświetlającą współrzędne `x` i `y`.
3. (**1 pkt**) Do klasy `Projectile` dodaj:
   (a) pozycję pocisku (pole chronione typu `Vector2D`),
   (b) kierunek ruchu pocisku (pole chronione typu `Vector2D`, musi być znormalizowane),
   (c) prędkość (pole chronione typu zmiennoprzecinkowego pojedynczej precyzji),
   (d) nazwę pocisku (pole chronione typu `char*`),
   (e) konstruktor inicjalizujący wszystkie pola parametrami (znormalizować kierunek!).
4. (**1 pkt**) W klasie `Projectile` metodę `Display` uczyń wirtualną (wyświetl pozycję, kierunek i prędkość), a `Frame` czysto wirtualną. Dodaj metodę `Shift` (zmieniającą pozycję zgodnie z kierunkiem i prędkością). Użyj operatorów `+=` oraz `*` klasy `Vector2D`.
5. (**2 pkt**) Utwórz klasy reprezentujące różne rodzaje pocisków: `ClusterBomb`, `ShotgunShell`, `Cluster`. Mogą się one wszystkie znaleźć w pliku `Projectile.h`. Każda powinna posiadać konstruktor przyjmujący pozycję na mapie i kierunek poruszania się. Wywołać odpowiednio konstruktor klasy bazowej. Jako prędkość podać: `ClusterBomb` $\leftarrow$ 2, `ShotgunShell` $\leftarrow$ 50, `Cluster` $\leftarrow$ 100. Jako nazwę podać dowolny sensowny napis. W klasie `ClusterBomb` powinien się znaleźć dodatkowy licznik inicjalizowany wartością 5.
6. (**1 pkt**) Metoda `Display` w klasach pochodnych `Projectile` powinna wyświetlać nazwę pocisku oraz wywoływać metodę `Display` klasy bazowej.
7. (**3 pkt**) Metoda `Frame` w klasach pochodnych `Projectile` zaimplementować w pliku `Projectile.cpp`. Ma ona wywołać metodę `Shift` klasy bazowej. Ponadto
   (a) dla klasy `ShotgunShell` zmniejszyć prędkość o 10, a gdy ta spadnie poniżej zera, wyświetlić "pocisk upadł" (usunąć obiekt z kolekcji pocisków),
   (b) dla klasy `Cluster` zmniejszyć prędkość o 5, a gdy ta spadnie poniżej zera wyświetlić "odłamek upadł" (usunąć obiekt z kolekcji pocisków),
   (c) dla klasy `ClusterBomb` zmniejszyć prędkość o 0,2 (jeżeli spadnie poniżej zera ustawić na zero) oraz zdekrementować licznik. Gdy wartość licznika spadnie poniżej zera wyświetlić: "Granat wybucha i rozpryskuje 4 odłamki", usunąć obiekt z kolekcji pocisków, dodać do kolekcji cztery obiekty klasy `Cluster` (pozycja taka jak granatu, który wybuchł, kierunek dowolny).

Sprawdzić działanie poprzez:

```
projectiles.Add(new ClusterBomb(Vector2D(0, 0), Vector2D(1, 1)));
projectiles.Add(new ShotgunShell(Vector2D(-10, -20), Vector2D(-10, -5)));
```