Laborator 8 – Funcții și clase template

Tema: Entities Spawn Engine

Autor: ing. Eduard Donea

Poveste

Daria este o dezvoltatoare de jocuri video aflată la început de drum. Ea dorește să creeze un sistem de generare (spawn) de entități pentru un joc de tip open-world. Sistemul trebuie să poată genera diferite tipuri de entități: inamici, personaje neutre, obiecte colectabile. Pentru a evita duplicarea codului de spawn și logica de administrare, Daria va folosi `template`-uri în C++.

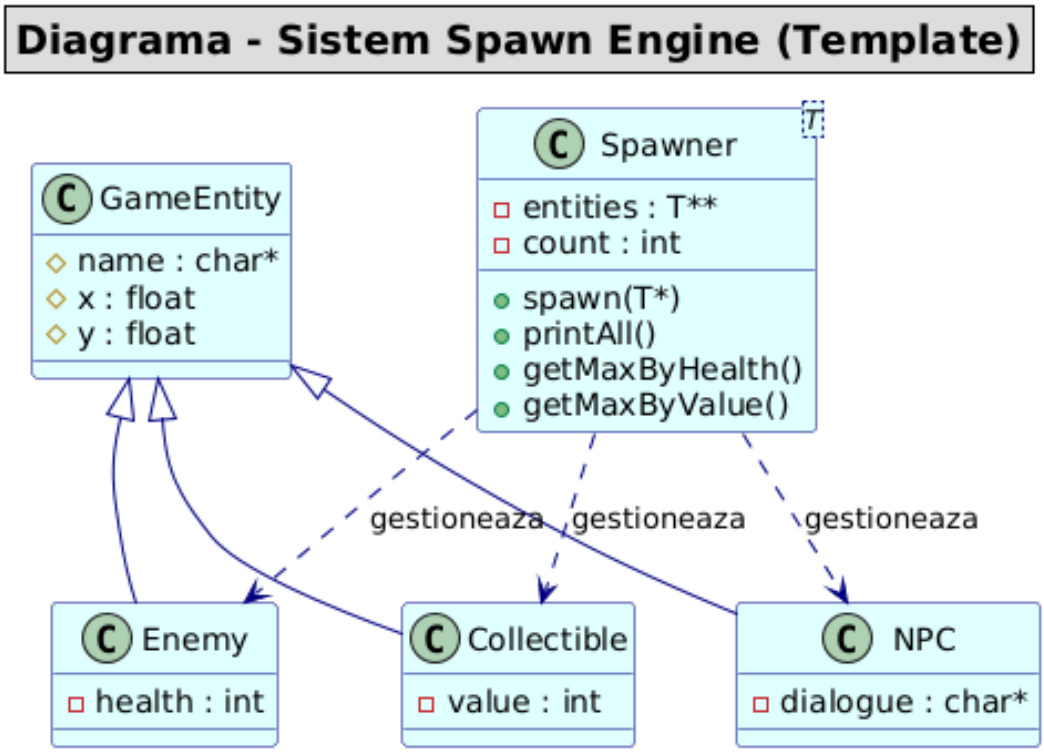
Scopul laboratorului

- introducerea programării generice cu **template-uri C++**;
- crearea unui sistem flexibil, reutilizabil, specific Game Development;
- lucrul cu moștenire simplă și componență;
- gestionarea memoriei și separarea fișierelor într-un proiect C++.

Structura fișierelor

- **GameEntity.h** – clasa de bază;
- **Enemy.h** – clasa derivată (inamici);
- **NPC.h** – clasa derivată (personaje neutre);
- **Collectible.h** – clasa derivată (obiecte colectabile);
- **Spawner.h** – clasa template pentru generarea entităților;

Figura 1 – Diagrama UML a sistemului Spawn Engine



În continuare aveți scheletele claselor. **Atenție: metodele trebuie implementate de voi acolo unde lipsesc. La template, aveți .h, iar la .cpp aveți doar un model de implementare pentru o metodă.**

Fișierul `GameEntity.h`

```h
#ifndef GAMEENTITY_H
#define GAMEENTITY_H

// Includeti aici bibliotecile necesare

class GameEntity {
protected:
    char* name;
    float x, y; // coordonate in joc

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `Enemy.h`

```h
#ifndef ENEMY_H
#define ENEMY_H

#include "GameEntity.h"

class Enemy : public GameEntity {
private:
    int health;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `NPC.h`

```h
#ifndef NPC_H
#define NPC_H

#include "GameEntity.h"

class NPC : public GameEntity {
private:
    char* dialogue;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `Collectible.h`

```h
#ifndef COLLECTIBLE_H
#define COLLECTIBLE_H

#include "GameEntity.h"

class Collectible : public GameEntity {
private:
    int value;

public:
    // TODO: Scieti aici tot ceea ce aveti nevoie.
};

#endif
```

Fișierul `Spawner.h`

```h
#ifndef SPAWNER_H
#define SPAWNER_H

#include "Enemy.h"
#include "Collectible.h"

template <class T>
class Spawner {
private:
    T** entities;
    int count;

public:
    Spawner();
    ~Spawner();

    void spawn(T* e);         // adauga o entitate
    void printAll() const;    // afiseaza toate entitatile

    T* getMaxByHealth();      // pentru Enemy
    T* getMaxByValue();       // pentru Collectible

};

#endif
```

Fișierul `Spawner.cpp` – model de implementare

```cpp
#include "Spawner.h"

template <class T>
Spawner<T>::Spawner() : count(0), entities(nullptr) {}

// Mereu sa aveti grija sa puneti "template <class T>" la
// inceput de orice metoda/constructor template.

// Nu uitati sa va scieti void test(), inclunzand inauntrul ei
// tot ce faceti in "int main()", pana la "return 0", nu includeti si asta.
```

Cerințe (10p)

1. (3p) Implementați clasele `GameEntity`, `Enemy`, `NPC`, `Collectible` completând metodele lipsă;
2. (3p) Implementați clasa template `Spawner`;
3. (2p) Testați funcțiile template în `main.cpp`;
4. (1p) Utilizați `getMaxByHealth()` și `getMaxByValue()`;
5. (1p) Eliberați memoria după utilizare și verificați cu `valgrind`.

Compilare si rulare

```
g++ -g *.cpp -o lab8
valgrind ./lab8
```

Bonus GitHub Classroom

Studenții pot obține până la **3p bonus** dacă încarcă toate laboratoarele până la finalul semestrului, respectiv **1p bonus** dacă încarcă mai mult de jumătate dintre ele.

Pași pentru încărcarea laboratorului:

1. Accesați cursul de pe OCW – Programare Orientată pe Obiecte, secțiunea **Resurse** → **Bonus GitHub laborator**.
2. Acolo veți găsi linkul **GitHub Classroom** pentru Laboratorul 8:

   `https://classroom.github.com/a/CeuVjhXY`

3. După acceptare, veți primi propriul repository, ex. `laborator-8-NumePrenume`.
4. Clonați repository-ul local:

```
git clone
↪  git@github.com:Laborator-POO-2025-2026/laborator-8-NumePrenume.git
cd laborator-8-NumePrenume
```

5. După ce ați adăugat fișierele, urmează comenzile:

```
git add .
git commit -m "Mesaj de commit"
git push origin main
```

Atenție: laboratoarele copiate între studenți sunt considerate nevalide, iar niciunul dintre cei doi participanți nu va primi bonusul. Repository-urile sunt monitorizate automat de sistemul GitHub Classroom.