**Ant Colony Simulation**

**Overview**

This project is a simulation of ant colonies built in C++. It models various ant species, their roles (Queen, Worker, Warrior), and their interactions within colonies. The goal of the simulation is to allow colonies to grow, compete, and survive. The simulation utilizes object-oriented principles and design patterns like Singleton, Factory, and Template for extensibility and modularity.

**Features**

* **Species Diversity**: Different ant species with unique attributes.
* **Colony Management**: Manage resources, workers, and warriors.
* **Combat Mechanics**: Inter-colony battles for survival.
* **Room Construction**: Build specialized rooms in colonies.
* **Dynamic Simulation**: Progress through simulation ticks, with automatic ant actions.

**Getting Started**

**Prerequisites**

Ensure you have the following installed on your system:

* A C++ compiler supporting C++11 or later.
* Windows OS for windows.h compatibility.

**Installation**

1. Clone the repository:

git clone <repository\_url>

1. Navigate to the project directory:

cd <repository\_folder>

1. Compile the code:

g++ main.cpp -o AntSimulation

**Running the Simulation**

Run the compiled executable:

./AntSimulation

**How to Use**

**Commands**

* **spawn X Y Species**: Create a new colony at coordinates (X, Y) of the specified species.
* **give ID Resource Amount**: Allocate resources (food/warriors/workers) to a colony.
* **tick [N]**: Progress the simulation by N ticks (default is 1).
* **summary ID**: Display the status of a colony by ID.
* **list**: Show all active colonies.
* **species**: Display available ant species and their attributes.
* **help**: Show the help menu.
* **quit**: Exit the simulation.

**Example Usage**

1. Start by spawning colonies:
2. spawn 5 5 FireAnts

spawn 10 10 Guardians

1. Add resources to colonies:
2. give 1 food 100

give 2 warrior 5

1. Advance simulation:

tick 10

1. Check colony status:

summary 1

1. List all colonies:

list

**Code Structure**

**Key Classes**

* **Ant**: Base class for all ants.
  + Derived classes: Queen, Worker, Warrior.
* **AntFarm**: Represents a colony, manages ants, rooms, and resources.
* **Room**: Abstract base for colony rooms.
  + Derived classes: RestingRoom, SpawningRoom.
* **AntMediator**: Facilitates interactions among ants.
* **Meadow**: Singleton class managing the entire simulation.
* **AntFactory**: Factory pattern for creating ants.

**Key Components**

1. **Species Attributes**:
   * Attributes like attack, defense, and harvesting bonuses are defined in SpeciesAttributes.
   * SpeciesInfo holds species-specific details.
2. **Colony Lifecycle**:
   * **Creation**: Spawn colonies using AntFarmBuilder.
   * **Management**: Allocate resources and monitor status.
3. **Combat System**:
   * Ants engage in battles, gaining bonuses upon victory.
   * Defeating a queen allows colony mergers.
4. **Rooms and Building**:
   * Specialized rooms like RestingRoom and SpawningRoom facilitate colony operations.

**Simulation Workflow**

1. Initialize the game using Meadow::getInstance().
2. Spawn colonies with strategic placement.
3. Allocate resources to grow and strengthen colonies.
4. Advance the simulation in ticks.
5. Monitor colony health, food reserves, and combat outcomes.
6. Achieve victory by eliminating other colonies.

**Technical Details**

**Design Patterns**

* **Singleton**: Meadow ensures only one simulation instance.
* **Factory**: AntFactory creates different types of ants.
* **Template**: AntFarm manages generic ant types.

**Dependencies**

* windows.h: Used for system-specific features like screen clearing and delays.

**Future Enhancements**

* Add graphical visualization for colony interactions.
* Implement additional ant roles (e.g., Scouts, Engineers).
* Extend room types with new functionalities.

**Acknowledgments**

Developed by **Shehr Bano (NUM-BSCS-23-11)** as a software engineering assignment. Inspired by nature's complexity and teamwork.