Miniature Kingdom Design Document

– A Complex Simulation Game with two distinct modes of player interaction

Miniature Kingdom will be a simulation sandbox game simulated at the level of individual agents. It will have two different systems of player interaction: one allowing the player to control individual agents and objects in the game world, and one giving the player more limited abstracted controls on the level of setting general rules for agents and systems to follow.

Overview

The game world of Miniature Kingdom will be small but richly populated with individual agents of various professions making up guilds interacting with each other and an environment of numerous interactable entities and systems in a small but complex “economy”.

Miniature Kingdom will have two distinct systems of player interaction, only one of which can be active at a time.

The more direct system of player control will give the player direct control over many elements of the simulation. They will place individual infrastructure buildings into the game world and control the individual agents directly.

The more abstracted player interaction system will give the player more indirect control. Instead of placing individual buildings they will set funding levels for various categories of infrastructure. Instead of controlling each agent directly they will set areas and degrees of focus for the guilds.

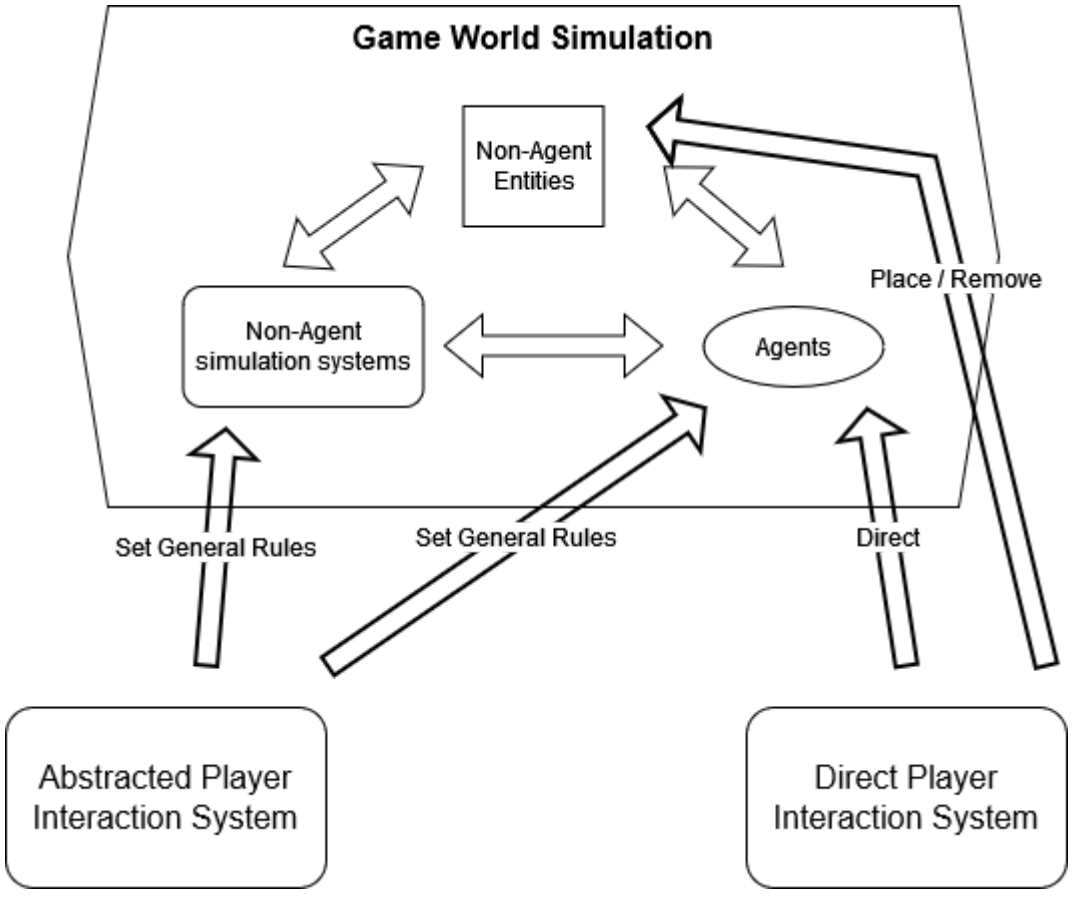


Table of Contents

Technology

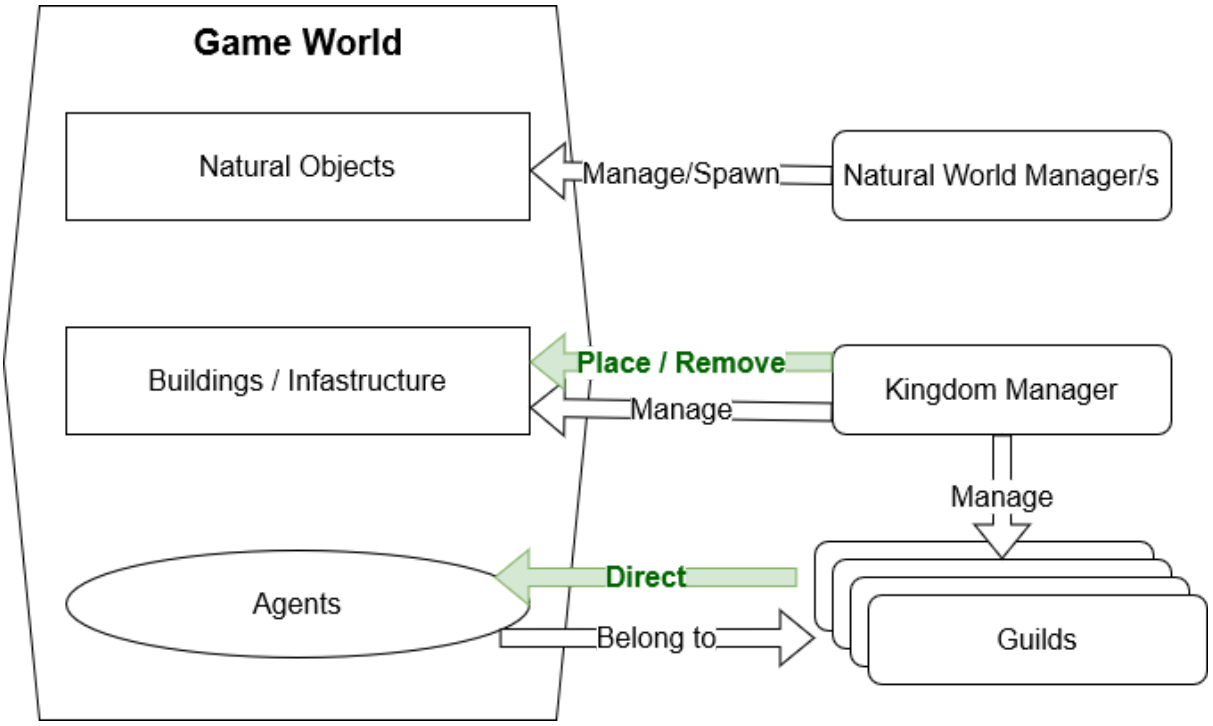
This game will be built using the Unity game engine. Unity is the engine the developer has the most experience with.

Classes

All classes will be written as C# scripts and extend Unity's MonoBehaviour class to allow them to be cleanly attached to Unity game objects.

Game World Simulation

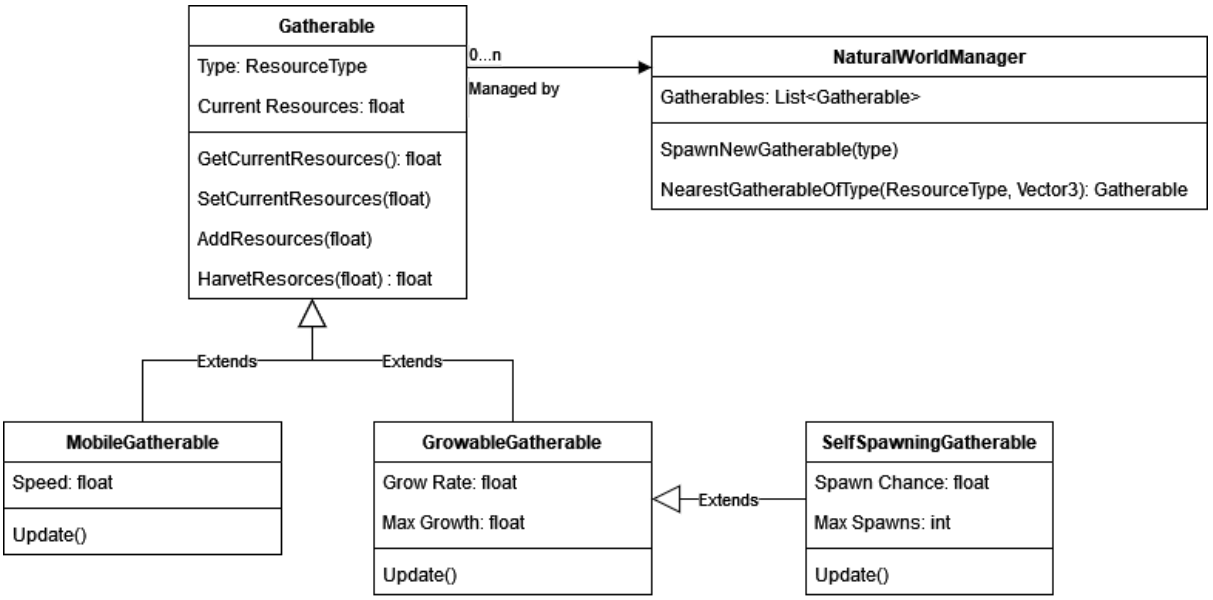
Below is a diagram showing the broad relationships between the various entities that will populate the game world and the systems that will control their behaviour.



Natural World Manager

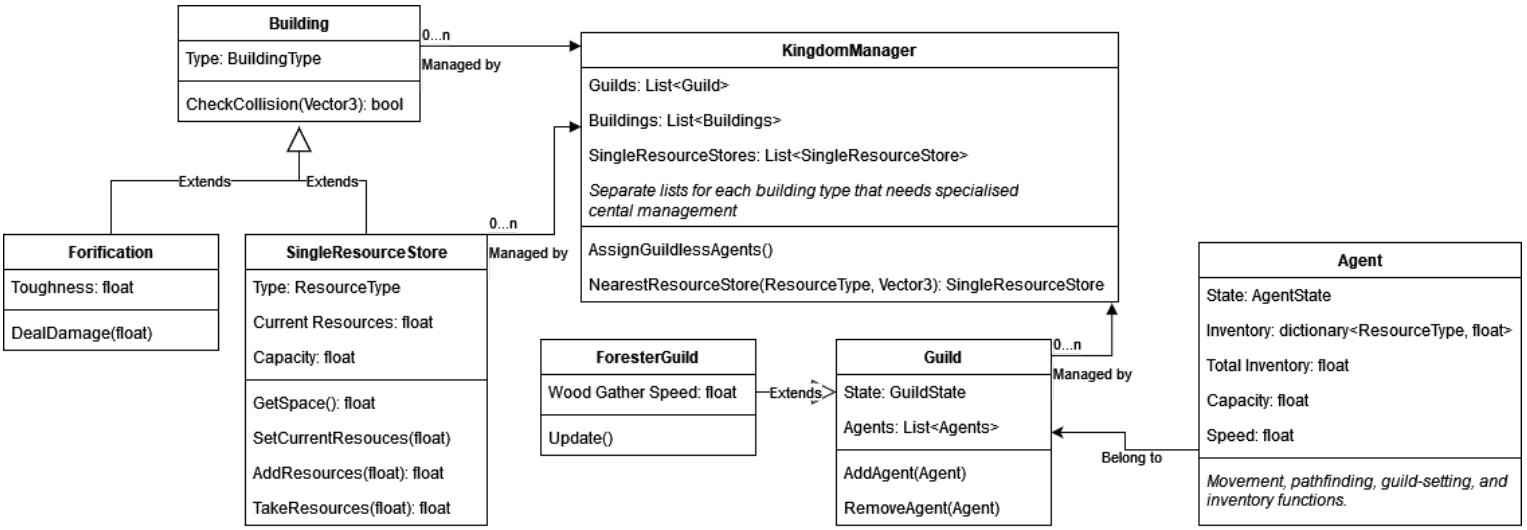
Mechanically, the natural world of this game will simply consist of a set of "Gatherables" from which agents can extract resources. Each Gatherable will hold a discrete amount of a single resource. There will be several derived versions of the Gatherable class including but not limited to MobileGatherables (animals), GrowableGatherables (such as trees that gain more wood as they age), and SelfSpawningGatherables that would spread copies of themselves nearby over time.

The Gatherables will be managed in the NaturalWorldManager which will provide a function for finding the nearest gatherable to a location that is of a certain resource type.



Kingdom Manager, Agents, and Guilds

The KingdomManager class will manage all the building and Guilds in the game. The guilds in turn will manage the Agents. Each Guild will be a unique derived class of the Guild base class and will contain the logic for Agents to complete a specific task. When no tasks need to be completed for a specific Guild it will free up any agents assigned to it.



The final game will have several more guilds and building types than the UML above but the same general relationship will be obeyed.

Agents

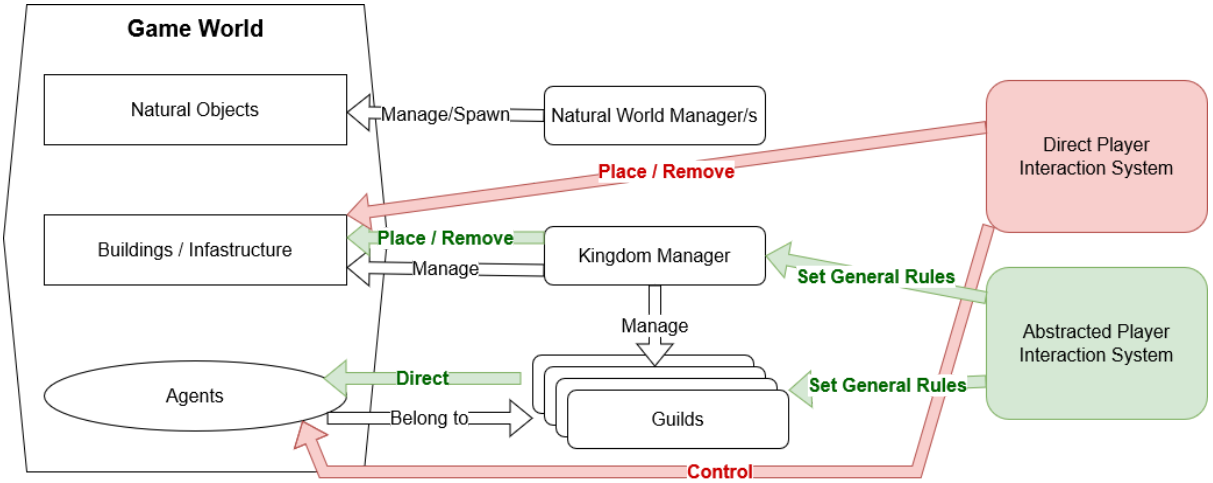
The agent class will contain the logic common to all agents (logic and functionality specifc to particular roles will be managed in the Guild sub-classes). This includes an inventory system, pathfinding, and hunger/exhaustion levels.

Buildings

Buildings will all be centrally managed in the KingdomManager. When buildings are initially placed (either by the player or by the KingdomManager) they will be inactive foundations until agents belonging to a “BuildersGuild” bring enough resources to it and spend enough time building it.

Player-Interaction Systems

The game will have two player-interaction systems. Only one will be active in any particular session and will be selectable on game start.



Abstracted

The abstracted player-interaction system will give the player no direct control over agents or buildings. Instead the player will be able to set general rules. The player will be able to set for example the proportion of agents assigned to guilds, the priority levels for developing various type of infrastructure, or the size of food rations.

The set of rules the player has control over will be closely watched during development. The list will need to stay short enough to keep the system simple but long enough and with enough individual impact to give the player a strong feeling of control.

Direct

The direct player-interaction system will provide a control scheme similar to most real time strategy (RTS) games. The player will be able to select and move individual agents and assign them to tasks by right-clicking on objects in the game world. The player will also have a menu of buildings that they can select from to place foundations into the scene.

When this system is active the KingdomManager and Guilds will serve a slightly more passive roll. The guilds will still determine the basic behaviour of Agents once assigned to a task but the assignment of agents to guilds will only be through the player assignment of tasks.