**Архитектура игры SummerGame**

**Абстрактная модель игры**

Решение SummerGame состоит из нескольких проектов:

* AbstractGameEngine
* GameEngine
* Geometry
* UserInterface
* SummerGame

Логическую часть решения содержат проекты: AbstractGameEngine, GameEngine, Geometry.

Подробней об AbstractGameEngine. Это проект, который содержит в себе интерфейсы и абстрактные классы для создания конкретных классов. Так как задумка с самого начала рассчитана на игру разновидности «стратегия» в этом проекте присутствуют такие сущности, как Administrant и Character.

**Administrant** является сущностью исполнителя каких-либо действий, то есть просто умеет совершать различные действия, а также имеет в себе сущность Side, которая отвечает за принадлежность его к чему или кому-либо, так как эта абстрактная модель, то мы не можем окончательно сказать по поводу того, будет ли принадлежность, какая она будет или не будет совсем.

**Character** же более конкретное представление участника игры, он имеет набор свойств, который также основан на сущности UnitProperties.

Для каждой из этих сущностей должно быть задано правило действия для того, чтобы в игре появились хоть какие-то движения или цели. Для этого введена сущность Action, которая явно говорит о своем значении.

**Action** имеет в себе цель исполнять любые запросы персонажей игры, поэтому существует параметризованный вид Action, чтобы в него можно было поместить любое действие с помощью делегата. Также Action имеет в себе состояние действия, закончилось ли оно, время исполнения, и, конечно, методы для исполнения действия в зависимости от делегата.

Чтобы персонажи могли что-то делать, им нужно где-то это делать, поэтому возникает прямая необходимость ввести сущность Area, которая будет просто отвечать за занятый участок карты каким-либо объектом, персонажем и так далее.

**Area** – это сущность занятой области и только в ней использованы сущности проекта Geometry. Причина по которой в абстрактной модели игры используются конкретные классы проекта Geometry интуитивна понятна. Нам нужны фигуры для описания области.

Конечно, нам в игре нужны предметы, поэтому была введена сущность **Item**. Ничего особенного она собой не представляет. Может лишь занимать какую-либо область.

Далее предполагалось, что персонажи будут общаться между собой, поэтому появилась сущность **Message**, она несет в себе какой-то Action, а также информацию о том, кто послал сообщение, и кому.

Так как игра разновидности «стратегия», то нужно иметь ввиду, что нам нужно объединять персонажей в группы, за это будет отвечать **Squad**.

На начальном этапе в абстрактной модели также существовала сущность Game, которая в итоге оказалась излишней и была исключена. После же была реализована, как вспомогательный класс в логике игры (проект GameEngine).

**Логика игры**

Игровая логика заключена и конкретизирована в проекте GameEngine, который содержит в себе различные классы, в том числе те, которые способствуют проведению игры, а не участвуют в ней.

В GameEngine описаны классы, реализующие абстрактную модель и различные взаимодействия, а также в ней присутствуют отдельные блоки конкретизации формул, расчетов игровых, и, конечно, алгоритмы и методы для обработки игровых событий.

Логика игры описывается в несколько блоков:

* Реализация персонажей (Character).
* Реализация Action, различных действий доступных персонажу.
* Реализация различных видов местности (Lands от Area).
* Реализация предметов для игры (Object от Item).
* Классы, отвечающие за формирование историй, которые передаются интерфейсу.
* Расширения и статический класс для Загрузки игры и сохранения.
* Класс Game, который отвечает за обсчет игрового времени и высылку историй интерфейсу.

Подробнее о составляющих GameEngine:

Реализация персонажей - создание конкретных классов войск для игры SummerGame.

Реализация Action - возможность передвигаться и атаковать.

Реализация местности - ограничение скорости передвижения по разным местностям.

Реализация предметов игры - аммуниция для боя, техника и прочее.

Формирование историй - создание набора действий на промежуток времени для игрового интерфейса, чтобы пользователь мог видеть происходящие события игровой модели.

Расширение и класс сериализатор - для загрузки и сохранения объектов карты, персонажей (войск), а также областей карты, местностей.

Game - содержит в себе принадлежность хода игроку, отсчет игрового времени и карту, карта представляет из себя набор персонажей (войск), предметов, местностей. В методе посчета нового момента игрового времени этот класс отсылает набор историй проекту UserInterface для интерпретации на экране пользователя.