Архитектура игры

Описание:

2D игра с видом сверху на unity, про пошаговые бои кораблями.

Поле:

Гексагональное, создано с помощью Grid, Tilemap, в элементе Grid есть скрипт Grid.cs, который отвечает за логическую реализацию сетки, в него мы передаем элементы Grid, Tilemap.

Юниты:

Это префабы кораблей, есть несколько типов кораблей, на каждый свой префаб. В каждом префабе также находится скрипт Ship.cs, в нем описаны характеристики корабля, действия описаны в скриптах ShipMovement.cs и ShipCombat.cs. Список характеристик: количество действий, количество здоровья, броня, скорость передвижения, дальность атаки пушками, урон от дальней атаки, урон от абордажа, принадлежность к команде (игрок или противник), направление (текущее направление корабля).

Ход игры:

Когда ход переходит к игроку, у него есть ограниченное количество действий (зависит от характеристики корабля). Он может выполнять любые действия, включая повторяющиеся, также есть кнопка завершить ход. После завершения хода ходит следующая команда (пока что простой ии для врага). Если у команды несколько кораблей, то завершение хода активирует следующий корабль, который ещё не походил на данном ходу. Логика ходов находится в скрипте TurnManager.cs.

Движение:

Когда игрок выполняет действие движение (нажимая кнопку M на клавиатуре),, и он перемещает корабль на число клеток равное скорости по прямой в сторону его направления. Движение описано в скрипте ShipMovement.cs.

Поворот:

Когда игрок выполняет действие поворота (нажимая кнопку Q или E на клавиатуре), корабль меняет направление на 60 градусов влево (Q) или вправо (E). Поворот описан в скрипте ShipMovement.cs.

Абордаж:

Если в соседней клетке от игрока есть вражеский корабль, игрок может выполнить действие абордажа (кнопка A на клавиатуре), далее игрок кликает на вражеский корабль. При этом сравниваются характеристики урона от абордажа двух кораблей, это различные кубики (например D6 - шестигранный кубик). Атакующий корабль кидает 3 своих кубика, защищающийся 2, тем самым разница выпавших значений наносится уроном по кораблю с наименьшим числом.

Пример: у атакующего корабля характеристика урон от абордажа = D6 (случайное число от 1 до 6), у защищающегося = D4. На 3 кубиках атакующего выпали значения 1, 5 и 3, в сумме 9. На 2 кубиках защищающегося 2 и 4, в сумме 6. Защищающийся корабль получил 3 урона.

Абордаж описан в скрипте ShipCombat.cs.

Стрельба:

Если в радиусе дальней атаки корабля есть вражеский корабль, игрок может выполнить действие стрельбы (кнопка S на клавиатуре), далее игрок кликает на вражеский корабль. Игрок смотрит на характеристику атаки, кидает кубики в соответствии с ней, сравнивает сумму с показателем брони защищающегося корабля. Если сумма на кубиках выше брони, разница наносится уроном по защищающемуся кораблю.

Пример: показатель атаки корабля = 2D6, игрок выкинул 3 и 5, показатель брони защищающегося корабля= 5, этот корабль получит 8 - 5 = 3 урона.

Про радиус атаки: Определить радиус атаки как все клетки в пределах дальности атаки, которые находятся в 4 боковых направлениях от корабля (т.е. не спереди и не сзади).
Стрельба описана в скрипте ShipCombat.cs. Алгоритм: Определить радиус атаки как все клетки в пределах дальности атаки, которые находятся в 4 боковых направлениях от корабля (т.е. не спереди и не сзади).

Начало и конец игры:

Спавнится корабль игрока (случайная клетка в левой трети поля), также 3 корабля противника (3 разные случайные клетки в правой трети поля). Начинается ход игрока. Игра кончается, когда у одной из команд кончились все корабли. Это описано в скрипте GameManager.cs.

Логика ходов противника (будущие версии):

Пока что за обе команды играет пользователь. В будущем же можно реализовать следующие идеи. Для каждого корабля противника создавать систему принятия

решений. Простейший вариант — Finite State Machine (Конечный автомат) с состояниями вроде Idle -> ChooseTarget -> Move -> Attack -> EndTurn.
Решения должны приниматься на основе факторов: расстояние до игрока, текущее здоровье, возможность атаковать и т.д. Это можно реализовать через оценочные функции (scoring functions) или, для более сложного ИИ, через дерево поведений (Behaviour Tree).

Возможные улучшения базовых механик:

Реализовать движение по следующим правилам:

1. у корабля есть направление, он может двигаться только по прямой (гекс спереди от направления) на любое количество клеток, которое не превышает его скорость передвижения.
2. один раз за действие движения корабль может повернуть (причем как в самом начале, то есть до прохождения клеток, как в середине, так и в конце после прохождения всех клеток), но только максимум на 60 градусов от своего направления (2 гекса по бокам от гекса с кораблем и гекса спереди от направления).
3. Корабль не может двигаться сквозь другие корабли.

Добавить при получении урона кораблем систему повреждения: атакующий игрок бросает 2D6 и сравнивает результат с таблицей:

2-7	ничего не происходит
8	Корабль вычитает один из бросков абордажа
9	Корабль вычитает один из бросков стрельбы
10	Корабль теряет одну скорость передвижения
11	Корабль теряет одну броню
12	Корабль теряет одно действие

Добавить спавн препятствий, через которые нельзя перемещаться и стрелять.

Обертка геймплейного процесса:

Рогалик с прогрессией внутри забега: карта с различными узлами, по которой игрок перемещается, каждый узел это бой или спец событие, где можно получать улучшения. В конце карты находится босс, победив которого игрок переходит на следующую карту. Проходя игру, становятся доступны новые уровни сложности, с большим числом врагов и их улучшенными характеристиками.