**Морской бой**.

В репозитории имеется два сценария.

Сценарий sea\_battle.py реализует игру между человеком и машиной. Машина делает случайные выстрелы.

Сценарий sea\_battle\_optimal.py реализует игру между человеком и машиной. Машина пытается следовать оптимальной стретегии.

Далее приводится описание кода.

**Состав модулей**.

sea\_battle\_definitions модуль, содержащий основные определения

sea\_battle\_interface модуль, реализующий интерфейс игры

sea\_battle\_classes модуль, содержащий классы, реализующие логику игры

sea\_battle\_simple модуль, реализующий игру, в которой машина делает случайные выстрелы

sea\_battle\_optimal модуль, реализующий игру, в которой машина следует оптимальной стратегии

**Классы sea\_battle\_classes.**

Класс Cell описывает клетку игрового поля. Имеет следующие булевские свойства.

cccupied клетка занята кораблем

hit в клетку произведен выстрел

Класс Ship описывает корабль. Имеет следующие свойства.

ndecks число палуб

lives число жизней (первоначально равно ndecks)

body список клеток, входящих в тело корабля

contour список клеток, входящих в контур корабля

direction ориентация корабля (горизнтальная или вертикальная)

Класс имеет следующие методы.

locate располагает корабль на поле

good\_location возвращает True, если расположение корабля является допустимым

Расположение корабля должно удовлетворять следующим условиям: клетки тела и контура корабля не должны быть заняты другим кораблем.

Корабль создается следующим образом: строится список variants всех возможных допустимых положений корабля, если список непуст, выбирается один из вариантов случайным образом, в противном случае возбуждается исключение ShipCreationError.

Класс Board описывает игровую доску. Имеет следующие свойства.

nships число кораблей

fleet список кораблей

hidden доска скрыта

field список клеток игрового поля

Класс имеет следующие методы.

add\_ship включает во флот уже созданный корабль

create\_fleet создает флот в следующем составе: один линкор (3 клетки), два крейсера (2 клетки), 4 эсминца (1 клетка)

draw выводит текущее состояние доски

shot реализует выстрел противника и возвращает результат (промах, попадание, корабль потоплен)

Класс Gamer описывает участника игры. Обладает свойствами ownboard и oppboard (ссылки на собственную доску и доску противника). Имеет следующие методы.

get\_target определяет цель для следующего выстрела

shot выстрел, производимый участником

move ход участника (может состоять из нескольких выстрелов)

Класс Man (наследник Gamer) описывает игрока-человека. Метод get\_target позволяет ввести координаты выстрела.

Класс Machine (наследник Gamer) описывает игрока-машину. Метод get\_target позволяет случайным образом выбрать цель выстрела из списка клеток, в которые высрел еще не производился (список not\_tried, после каждого выстрела выбранная цель удаляется из списка not\_tried).

Класс Game описывает игру. Включает человека-участника и машину-участника (man и machine) и две игровые доски участников (man\_board и machine-board).

**Модуль sea\_battle\_optimal**.

Машина следует оптимальной стратегии, реализованной классом IntelligentMachine (наследник класса Machine, у которого переопределены методы get\_target и shot).

В случае попадания во вражеский корабль составляется список целей для следующего выстрела cells\_to\_try. Если произошло одно попадание (но корабль противника не потоплен), то в спискок включаеются соседние (но не диагональные) клетки так, как показано на рис. 1 (X - пораженная клетка, С - кандидат для следующей цели).

С

C X C

C

Рис.1. Выбор цели при одном попадании.

Если произошло попадание в две соседние клетки (но корабль противника не потоплен), то это означает, что поражен линкор противника (3 клетки), и выбор делается в зависимости от ориентации так, как показано на рис. 2.

С

C X X C X

X

C

Рис.2. Выбор цели при двух попаданиях.

Цель должна удовлетворять следующим условиям.

1. Цель не должна выходить за пределы игрового поля.
2. Цель не должна быть клеткой, в которую уже произведен выстрел.
3. Цель не должна быть клеткой, входящей в тело или контур уже потопленного корабля противника.

Выполение этих условий проверяется методом add\_cell\_to\_try.

Метод IntelligentMachine.shot реализован следующим образом.

1. Если произошло попадание в корабль противника (но корабль не потоплен), то в список cells\_to\_try включаются клетки, удовлетворяющие описанным выше условиям так, как это показано на рис. 1, в список пораженных клеток hits влючается пораженная клетка (атрибут last).
2. Если произошло попадание в две соседние клетки (hits уже содержит одну клетку), но корабль не потоплен, то в список cells\_to\_try включаются клетки, удовлетворяющие описанным выше условиям так, как это показано на рис. 2.

Выбор цели для выстрела делается следующим образом.

1. Если список cells\_to\_try непуст, то делается случайный выбор из этого списка, выбранная цель из списка удаляется.
2. Если cells\_to\_try пуст, то делается случайный выбор из списка not\_tried.

Если в результате очередного выстрела корабль противника потоплен, то выполняется следующее.

1. Списки hits и cells\_to\_try очищаются.
2. Все клетки контура потопленного корабля исключаются из списка not\_tried.

**Примечание**. Оптимальная стратегия значительно повышает шансы машины на выигрыш. Возможно, что описанная стратегия может быть усовершенствована, если учесть то обстоятельство, что число кораблей и условия их расположения делают весьма высокой вероятность расположения кораблей вдоль границы поля.