Задача 1 - 2

См код helicopters_fight выполнен на Python 3
Модель игры:
вертолет описывается набором параметров, каждый влияет или на атаку или на защиту.
Защитные параметры(armor stats): Speed, Armor, life
Атакующие(war stats): damage, capacity, rapidity

Каждый вертолет оценивается по суммарному показателю защиты и атаки.
Расчёт по формулам: x_armor = speed * 0.1 + armor * 0.2 + life * 0.7
y_war = damage * 0.7 + capacity * 0.2 + rapidity * 0.1

На двумерной плоскости (показатель защиты от показателя атаки) для каждого вертолёта изображается соответствующая точка. Вокруг точки рисуется красный круг - это параметр отличия вертолета (Dmax).. Если в пересечении кругов попадают сами точки вертолётов - они похожи.

Модель генерирует массив вертолетов с с рандомными параметрами, и потом сталкивает по два вертолета друг с другом, выигрывает один, второй либо ломается либо уничтожается. В каждой битве победителю даются очки. В конце в живых остается только один, и выводится таблица с очками.
События игры:
Ammunition is ended: no one wins!!!!! Если capacity обоих вертолётов < 1
Helicopter fires: вертолёт атакует и результат либо **Helicopter miss!!!**, либо **Helicopter succeed!!!** при этом отсчитывается урон второму вертолёту и пока armor > 0 и armor > repair вертолёт считается выжившим и бой продолжается **Helicopter survived: life left: {}; ammo left: {}.** Если после урона вертолёту его очки armor <= 0, этот вертолёт считается **Helicopter killed,** или если очки armor <= очков repair, это вертолёт **Helicopter fall down,** соответственно вертолёт нанёсший урон считается **Helicopter wins!!!** и ему присваиваются очки **Helicopter gets {} points.** После каждой битвы подсчитывается и стоимость, необходимая для починки и покупки ammunition **Helicopter {}:\nprice for refilling ammo: {}\nprice for repairing: {}**

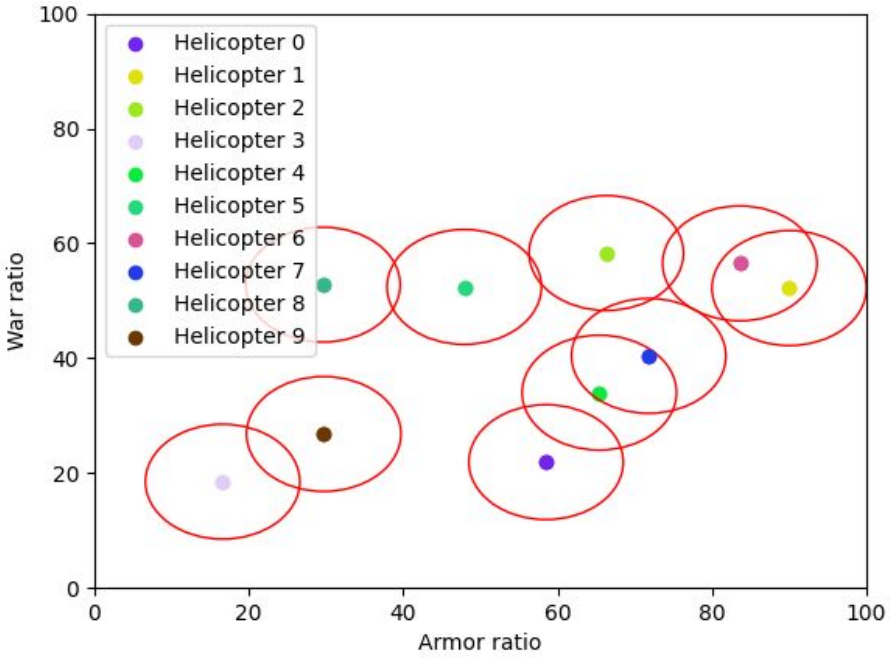
Количество вертолетов и параметр отличия можно менять

Все строится на генерации случайности

Вариант выгруженных данных:
Таблица параметров Вертолётов, столбцы - номера вертолётов

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
speed	84	92	38	42	92	80	70	37	96	66
armor	23	96	22	55	64	63	89	71	37	28
life	65	88	83	2	62	39	84	77	18	25
x_sum_armor	**58.5**	**90**	**66.3**	**16.6**	**65.4**	**47.9**	**83.6**	**71.8**	**29.6**	**29.7**
damage	2	52	45	9	33	52	68	37	43	7
capacity	84	30	89	52	30	43	3	53	73	96
rapidity	37	98	90	18	49	74	83	39	81	27
y_sum_war	**21.9**	**52.2**	**58.3**	**18.5**	**34**	**52.4**	**56.5**	**40.4**	**52.8**	**26.8**

Ниже в представленном графическом виде хорошо видно, что пары вертолётов (1 - 6) и (4 - 7) - между собой похожи



Теперь проводится DethMatch, пример раунда выглядит таким образом:

```
!!! ROUND 9 FIGHT !!!
!!!!!!FIGHT!!!!!!FIGHT!!!!!!FIGHT!!!!!

Helicopter 5 VS Helicopter 2

Helicopter 5 fires:
Helicopter 5 miss!!!
Helicopter 2 survived: life left: 13.001999999999999;  ammo left: 89
Helicopter 5 fires:
Helicopter 5 miss!!!
Helicopter 2 survived: life left: 13.001999999999999;  ammo left: 89
Helicopter 5 fires:
Helicopter 5 succeed!!!
Helicopter 2 killed

!!!!!!!!!!!!!!!!!!
Helicopter 5 wins!!!
Helicopter 5 gets 2 points
======================================================================
Helicopter 5:
price for refilling ammo: 300
price for repairing: 13

Helicopter 2:
price for refilling ammo: 0
price for repairing: 69
```

Таблица итогов:

```
======= !!!!!! Helicopter 5 Wins Deathmatch !!!!! =======

Score table:

Helicopter 0 score: 0
Helicopter 1 score: 5
Helicopter 2 score: 1
Helicopter 3 score: 0
Helicopter 4 score: 3
Helicopter 5 score: 7
Helicopter 6 score: 4
Helicopter 7 score: 0
Helicopter 8 score: 0
Helicopter 9 score: 0
```

Задача 3

Игровая механика и баланс очень нетривиальные вещи, и все коэффициенты ещё неоднократно корректируются при тестировании игры.. Долго думала от чего отталкиваться, оставить ли набор опыта постоянной величиной, связывать ли его с временем или нет, отталкиваться ли от того, что все противники должны быть побеждены, или должна быть захвачена какая-то база. В итоге набросала такую таблицу, но конечно, тут всё обсуждаемо.

Итак, я решила взять рост общего опыта как квадратичную функцию, а окончание уровня, когда все противники повержены, тогда время уровня можно нормировать на единицу и за оставшиеся минуты, если уровень пройден быстрее, начислять бенефит игроку

Здесь:
XP - полученный опыт игрока, после которого переходит на новый уровень,
level_user - прокачанный уровень персонажа
max_level_mob - максимальный уровень противника
quantity_of_mobs - минимальное количество противников
mobs - условное количество противников разного уровня и сколько опыта они приносят
delta_XP - разница опыта которую нужно набрать для следующего уровня игры
go - сколько попыток игрок сделал, пытаясь пройти уровень

	XP	level_user	max_level_mob	quantity_of_mobs	mobs	delta_XP	go
1	0	3	1	2	500*2	1000	1
2	1000	3	2	3	1000*1 + 500*2	2000	1
3	3000	4	2	4	1000 *1 + 500*4	3000	1
4	6000	4	3	5	1500*1 + 1000*1 +500*3	4000	2
5	10000	4	3	6	1500*1 + 1000*2 + 500*3	5000	2
6	15000	5	3	7	1500*2 + 1000*1 +500*4	6000	2
7	21000	5	4	9	2000*1 + 1000*2 + 500 * 6	7000	3
8	28000	6	4	6	2000*2 + 1000*4	8000	2
9	36000	6	4	7	2000*3 +1500*1 +500*3	9000	3
10	45000	6	4	8	2000*4 + 500*4	10000	1
11	55000	7	5	7	2500*2 + 1500*2 + 1000*3	11000	2
12	66000	7	5	10	2500*1 + 1500*4 +1000*4+ 500	12000	4
13	78000	7	5	6	2500*3 + 2000*2 + 1500	13000	1
14	91000	8	5	8	2500*4 + 1500*3 +1000*1	14000	2
15	105000	8	6	8	3000*2 + 2000*3 +1000*3	15000	3

Исходя из условия, что время уровня фиксированное N, уровни как правило проходятся быстрее лимита, но некоторые потребуют от игрока несколько попыток, поэтому в среднем всё равно на все уровни должно уйти 15 * N времени

Задачи 4, 5, 6 - **выполнены в Jupyter Notebooks, ибо так проще было по ходу выполнения максимально передавать свои мысли...**

P.S. ходят слухи, что никогда не прекращаются ярые обсуждения баланса игры между гейм дизайнерами, аналитиками и разработчиками, пытающимися при разработке брызжа слюнями друг на друга докопаться до истины как сделать лучше… но я не очень скиллованная, так что готова поначалу доверяться старшим, перенимая опыт :)