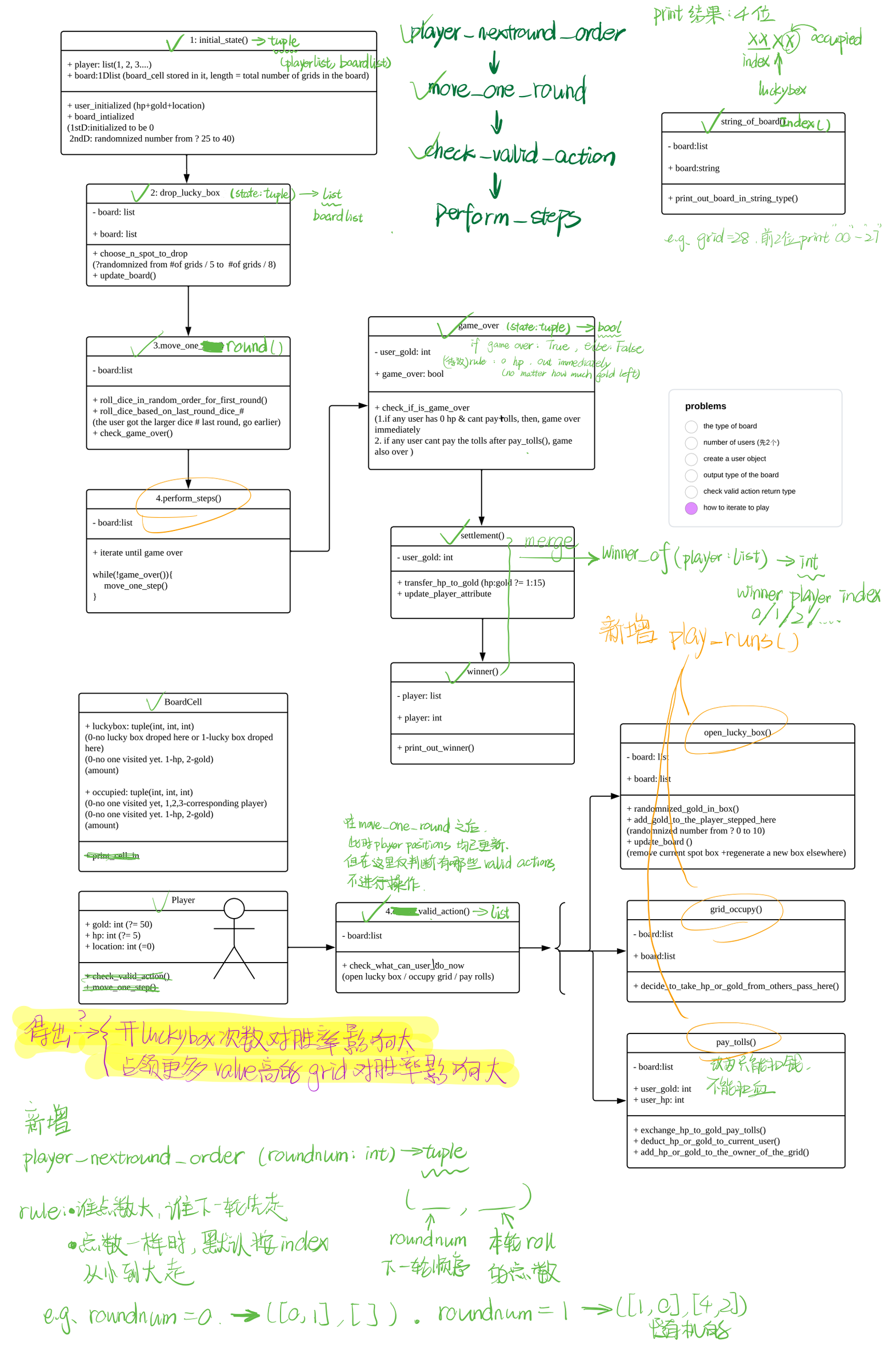
1: initial_state() → tuple
(playerlist, boardlist)
+ player: list(1, 2, 3....)
+ board:1Dlist (board_cell stored in it, length = total number of grids in the board)
+ user_initialized (hp+gold+location)
+ board_intialized
(1stD:initialized to be 0
2ndD: randomnized number from ? 25 to 40)
2: drop_lucky_box (state:tuple) → list
boardlist
- board: list
+ board: list
+ choose_n_spot_to_drop
(?randomnized from #of grids / 5 to #of grids / 8)
+ update_board()
3.move_one_round()
- board:list
+ roll_dice_in_random_order_for_first_round()
+ roll_dice_based_on_last_round_dice_#
(the user got the larger dice # last round, go earlier)
+ check_game_over()
4.perform_steps()
- board:list
+ iterate until game over
while(!game_over()){
move_one_step()
}
game_over (state:tuple) → bool
if game over: True, else: False
- user_gold: int
+ game_over: bool
(特效)rule: 0 hp, out immediately
(no matter how much gold left)
+ check_if_is_game_over
(1.if any user has 0 hp & cant pay tolls, then, game over immediately
2. if any user cant pay the tolls after pay_tolls(), game also over)
settlement()
merge
- user_gold: int
+ transfer_hp_to_gold (hp:gold ?= 1:15)
+ update_player_attribute
winner()
- player: list
+ player: int
+ print_out_winner()
winner_of(player:list) → int
winner player index
0/1/2/....
problems
the type of board
number of users (先2个)
create a user object
output type of the board
check valid action return type
how to iterate to play
player_nextround_order
move_one_round
check_valid_action
perform_steps
print 结果：4位
xx x(x) occupied
index
luckybox
string_of_board(index())
- board:list
+ board:string
+ print_out_board_in_string_type()
e.g. grid=28，前2位print "00"~"27"
新增 play_runs()
BoardCell
+ luckybox: tuple(int, int, int)
(0-no lucky box droped here or 1-lucky box droped here)
(0-no one visited yet. 1-hp, 2-gold)
(amount)
+ occupied: tuple(int, int, int)
(0-no one visited yet, 1,2,3-corresponding player)
(0-no one visited yet. 1-hp, 2-gold)
(amount)
~~print_cell_in~~
Player
+ gold: int (?= 50)
+ hp: int (?= 5)
+ location: int (=0)
~~+ check_valid_action()~~
~~+ move_one_step()~~
在move_one_round之后，
此时player positions 均已更新，
但在这里仅判断有哪些 valid actions，
不进行操作。
4.check_valid_action() → list
- board:list
+ check_what_can_user_do_now
(open lucky box / occupy grid / pay rolls)
open_lucky_box()
- board: list
+ board: list
+ randomnized_gold_in_box()
+ add_gold_to_the_player_stepped_here
(randomnized number from ? 0 to 10)
+ update_board ()
(remove current spot box +regenerate a new box elsewhere)
grid_occupy()
- board:list
+ board:list
+ decide_to_take_hp_or_gold_from_others_pass_here()
pay_tolls()
- board:list
+ user_gold: int
+ user_hp: int
或者只能扣钱，
不能扣血
+ exchange_hp_to_gold_pay_tolls()
+ deduct_hp_or_gold_to_current_user()
+ add_hp_or_gold_to_the_owner_of_the_grid()
得出？→ 开luckybox次数对胜率影响大
占领更多value高的grid对胜率影响大
新增
player_nextround_order (roundnum: int) → tuple
rule:•谁点数大，谁下一轮先走
•点数一样时，默认按index从小到大走
(__, __)
roundnum 下一轮顺序
本轮roll的点数
e.g. roundnum=0. → ([0,1],[]). roundnum=1 → ([1,0],[4,2])
也有机会

NN design considerations: Activation functions

Sign $(-1, +1)$

Heaviside $(0, 1)$

Tanh $(-1, 1)$

Sigmoid $(0, 1)$

Relu $(0, \infty)$

prob dist

$-\infty, \infty$