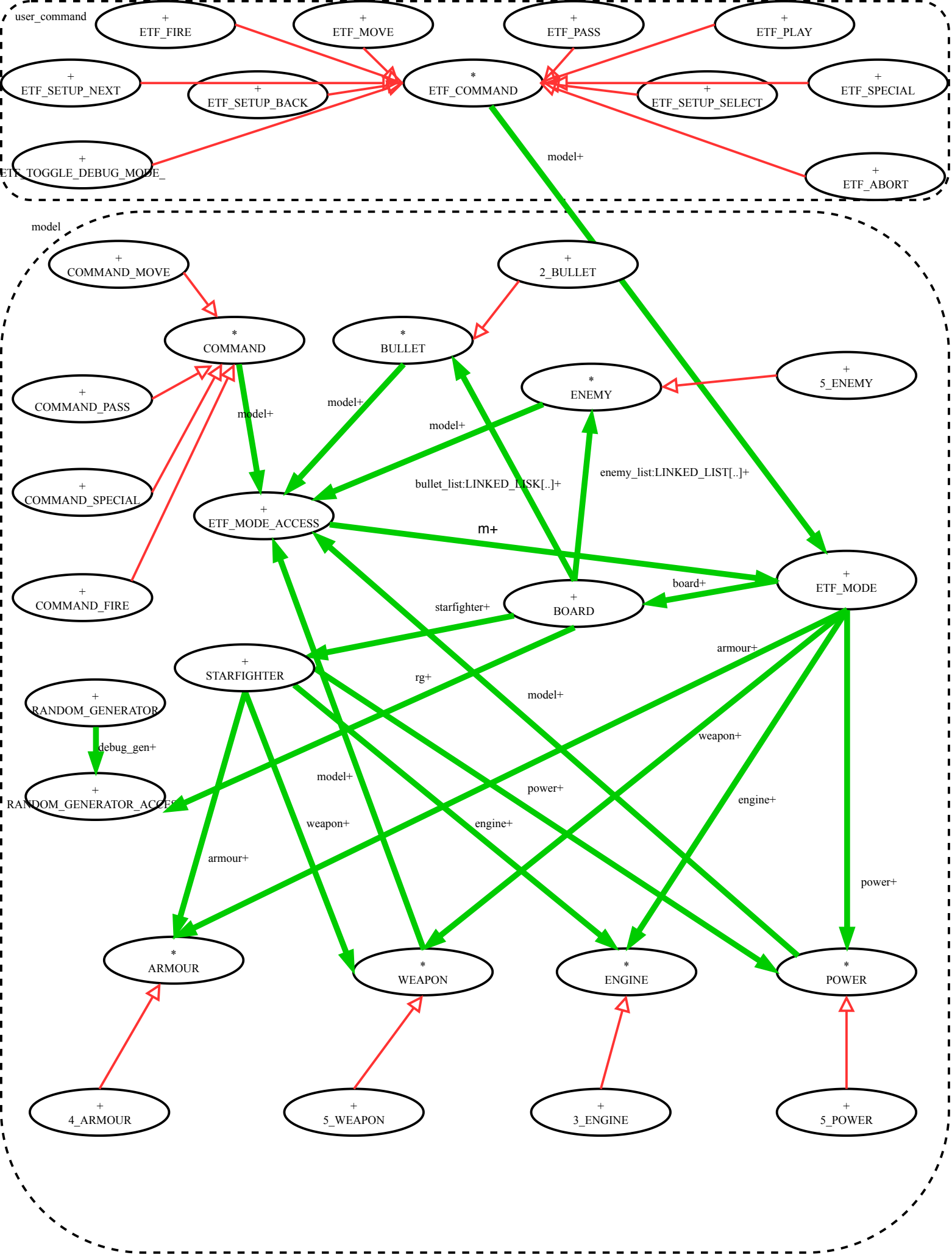
EECS3311
2020-FALL
YINGCHEN LIU
lyc99


user_command
+ ETF_FIRE
+ ETF_MOVE
+ ETF_PASS
+ ETF_PLAY
+ ETF_SETUP_NEXT
+ ETF_SETUP_BACK
* ETF_COMMAND
+ ETF_SETUP_SELECT
+ ETF_SPECIAL
+ ETF_TOGGLE_DEBUG_MODE_
model+
+ ETF_ABORT
model
+ COMMAND_MOVE
+ 2_BULLET
* COMMAND
* BULLET
+ 5_ENEMY
+ COMMAND_PASS
* ENEMY
model+
model+
model+
+ COMMAND_SPECIAL
bullet_list:LINKED_LISK[..]+
enemy_list:LINKED_LIST[..]+
+ ETF_MODE_ACCESS
m+
+ ETF_MODE
+ COMMAND_FIRE
board+
starfighter+
+ BOARD
armour+
+ STARFIGHTER
rg+
model+
+ RANDOM_GENERATOR
weapon+
debug_gen+
model+
power+
engine+
+ RANDOM_GENERATOR_ACCES
weapon+
engine+
armour+
power+
* ARMOUR
* WEAPON
* ENGINE
* POWER
+ 4_ARMOUR
+ 5_WEAPON
+ 3_ENGINE
+ 5_POWER

BOARD+

feature -- set board
make+(row:INTEGER;column:INTEGER;n1:INTEGER;n2:INTEGER;n3:INTEGER;
n4:INTEGER;n5:INTEGER;star:STARFIGHTER)
-- this function use to set up board

feature -- set game over
set_game_over+
-- set game over

feature -- turn
turn+(a:COMMAND)
-- do the 7 turn in order

feature -- set different action information
set_friendly_bullet_action+(information:STRING)
--this function use to change the friendly bullet action information

set_enemy_bullet_action+(information:STRING)
--this function use to change the enemy bullet action information

set_starfighter_action+(information:STRING)
--this function use to change the starfighter action information

set_enemy_action+(information:STRING)
--this function use to change the enemy action information

set_natural_enemy_action+(information:STRING)
--this function use to change the natural enemy action information

feature -- in board
bullet_in_board+(n:INTEGER):BOOLEAN
--this function use to test bullet is in board or not

bullet_in_board+(n:INTEGER):BOOLEAN
--this function use to test enemy is in board or not

feature{NONE} -- action of different act
friendly_projectiles_act+
--the action for friendly bullet

enemy_projectiles_act+
--the action for enemy bullet

starfighter_act+
--the action for starfighter

enemy_vision_update+
--the action for enemy vision update

enemy_act+
--the action for enemy

enemy_spawn+
--the action for create enemy

feature -- information
enemy_information+:STRING
--this function use to return enemy information

projectiles_information+:STRING
--this function use to return bullet information

five_action_information+:STRING
--this function use to return five different action information

feature -- insight
in_sighte+:BOOLEAN
--this function use to test enemy is in starfighter vision or not

feature{NONE} -- score
set_score+
--this function use to set score value

calculate_score+:INTEGER
--this function use to calculate score

platinum_calculate_function+:INTEGER
--this function use to calculate 3 time score

diamond_calculate_function+:INTEGER
--this function use to calculate 4 time score

feature -- out
out+
--return the message

starfighter

STARFIGHTER+

feature -- set straighter
make+(r:INTEGER;c:INTEGER;w:WEAPON;a:ARMOUR;e:ENGINE;p:POWER)
-- this function use set up starfighter

feature -- max health
max_health+:INTEGER
-- this function use to calculate max health

feature -- max energy
max_energy+:INTEGER
-- this function use to calculate max energy

feature -- armour
value_armour+:INTEGER
-- this function use to calculate starfighter's value of armour

feature -- regen
regen_health+:INTEGER
-- this function use to calculate starfighter's value of regen health

regen_energy+:INTEGER
-- this function use to calculate starfighter's value of regen energy

feature -- vision
vision_value+:INTEGER
-- this function use to calculate starfighter's value of vision

feature -- move
move_value+:INTEGER
-- this function use to calculate starfighter's value of move

move_cost+:INTEGER
--this function use to calculate starfighter's value of move cost

feature -- set
set_health+(h:INTEGER)
--this function use to change current health

set_energy+(e:INTEGER)
--this function use to change current energy

set_move+(n_row:INTEGER;n_column:INTEGER)
--this function use to change starfighter to new position

feature -- information
starfighter_information+:STRING
--this function use to return starfighter initial information

Section: Enemy Actions

My design idea is to first create an ENEMY deferred class. Then create five corresponding ENEMY classes. Each ENEMY class will have three different action categories.(Preemptive Action, Action when Star-fighter is not seen and Action when Star-fighter is seen). Each Preemptive Action will also have four different action categories(preemptive_action_fire, preemptive_action_special, preemptive_action_move and preemptive_action_pass). Whenever special command are used(Like Fire). Then will run stat-fighter's action. When the action of the star-fighter is finished, it will check all the existing enemies to see if there is preemptive action. If so, execute the corresponding command. Then update enemy vision. Now we come to the enemy actions. Enemy actions start from the first enemy and continue to the last. First check if the enemy has been destroyed (is it on the board). If it is destroyed, skip it, if not, enter its corresponding class. The first step is to check if it is seen by the star-fighter and carry out the corresponding instructions(Action when Star-fighter is not seen or Action when Star-fighter is seen). Then take the corresponding actions according to the instructions. Next I will give an example.

The example is ENEMY_GRUNY

```
preemptive_action_pass
do
  health:=health+10
  current_health:=current_health+10
  across 1 |..| model.m.board.enemy_list.count is enemy_n loop
     if
        model.m.board.enemy_list[enemy_n]=current
     then
        model.m.board.set_enemy_action(model.m.board.enemy_action+"%N    A Grunt(id:"+enemy_n.out+") gains 10 total health.")
     end
  end
end

preemptive_action_move
do

end

preemptive_action_special
do
  health:=health+20
  current_health:=current_health+20
  across 1 |..| model.m.board.enemy_list.count is enemy_n loop
     if
        model.m.board.enemy_list[enemy_n]=current
     then
        model.m.board.set_enemy_action(model.m.board.enemy_action+"%N    A Grunt(id:"+enemy_n.out+") gains 20 total health.")
     end
  end
end
```

First, when the user enters the pass command, GRUHT will run the function of preemptive_action_pass. Similarly, when the user enters the special command, GRUHT will run the function of preemptive_action_special. First we will give this Grunt the corresponding instructions(add current health and max health). Then look for its id and output the corresponding information.

```
    action_when_starfighter_is_not_seen
    local
        enemy_bullet:BULLET_ENEMY
        current_column:INTEGER
    do
      enemy_infront:=False
      message:=""
      from
        current_column:=column
        old_column:=column
      until
        current_health<=0 or current_column=column-2 or enemy_infront or current_column>model.m.board.board_column or current_column<=0
      loop
        if
          not across model.m.board.enemy_list is e_n some e_n.row=row and e_n.column=current_column-1 end
        then
          current_column:=current_column-1
        else
          enemy_infront:=True
        end
        if
          not (current_column>model.m.board.board_column or current_column<=0)
        then
          collided_with_bullet(current_column,enemy_name)
          collided_with_starfighter(current_column)
        end
      end

      column:=current_column
      across 1 |..| model.m.board.enemy_list.count is enemy_n loop
      if
        column=old_column
      then
        enemy_stay_information(enemy_n,old_column,model.m.board.enemy_list[enemy_n].enemy_number)
      else
        enemy_move_information(enemy_n,old_column,model.m.board.enemy_list[enemy_n].enemy_number)
      end
      end
    model.m.board.set_enemy_action (model.m.board.enemy_action+message)
  if
    column>=1
  then
    create{BULLET_ENEMY}enemy_bullet.make(row, column-1, 0, -1, 15, 4)
    model.m.board.bullet_list.extend(enemy_bullet)
    model.m.board.set_enemy_action(model.m.board.enemy_action+"%N      A enemy projectile(id:-"+model.m.board.bullet_list.count.out+") ")
    model.m.board.set_enemy_action(model.m.board.enemy_action+"spawns at location ["+model.m.board.row_letter[row].out+","+(column-1).out+"].")
    if
      enemy_bullet.row>=1 and enemy_bullet.row<=model.m.board.board_row and enemy_bullet.column>=1 and enemy_bullet.column<=model.m.board.board_column
    then
      message:=""
      collided_with_bullet(current_column,enemy_name)
      enemy_collided_with_enemy_bullet(enemy_bullet)
      collided_with_starfighter(current_column)
      model.m.board.set_enemy_action (model.m.board.enemy_action+message)
    end
  end
end
```

After finishing preemptive action, we have to run Action when Star-fighter is not seen or Action when Star-fighter is seen). Because these two actions are roughly the same, only the data is different, so I will only explain one of them here. Because there may be enemies, bullets or star-fighter on the way. So I used loop to move step by step. If there is an enemy in front, it will stop behind the enemy. If not, it will continue to move until the instruction is completed. If Grunt is not

out of bounds, it will judge whether his movement will hit a bullet or star-fighter every time. After the movement is completed, the current Grunt coordinates will be updated. If the new coordinates are the same as the old coordinates, the stay information is output, and the opposite is the move information. If Grunt is still alive and present after moving, a corresponding bullet will be created. Then the corresponding bullet information will be output, and it will be detected when it collides with the enemy, star-fighter, star-fighter's bullets or enemy bullets. If it collides, it will make corresponding instructions and output the corresponding information.

• Information Hiding (what is hidden and may be changed? what is not hidden and stable?)

Yes, my enemy action design and implementation is satisfies Information Hiding. Enemy action function in Board class. And only the turn function in the Board class can call this function. Other classes cannot use or modify enemy. So enemy action is hidden, and it may be changed when enemies are destroyed, or new enemies join. turn function is not hidden, but it is stable because only specific 4 instructions(fire, pass, move and specials) can use it.

• Single Choice Principle

No, my enemy actions design and implementation is not satisfies the Single Choice Principle. Because every enemy action must consider the problem of collision, and some also need to create bullets. So it didn't just do one thing

• Cohesion

No, my enemy actions design and implementation is not satisfies the Cohesion. Because it has to do more than one function. And it also considers collision issues and launching bullets.

• Programming from the Interface, Not from the Implementation

Yes, my enemy actions design and implementation is satisfies the Programming from the Interface, Not from the Implementation. First create an ENEMY (deferred class) and create some required deferred functions(such as preemptive action) and attributes(such as health and energy). Then five different enemy will inherit ENEMY. Then run their own functions.

Section: Scoring of Star-fighter

```
calculate_score:INTEGER
do
  if
    orb_list.count=0
  then
    Result:=0
  else
    from
      orb_position:=1
    until
      orb_position>orb_list.count
    loop
      if
        orb_list[orb_position]~"bronze"
      then
        Result:=Result+1
        orb_position:=orb_position+1
      elseif
        orb_list[orb_position]~"silver"
      then
        Result:=Result+2
        orb_position:=orb_position+1
      elseif
        orb_list[orb_position]~"gold"
      then
        Result:=Result+3
        orb_position:=orb_position+1
      elseif
        orb_list[orb_position]~"platinum"
      then
        Result:=Result+platinum_calculate_function
      elseif
        orb_list[orb_position]~"diamond"
      then
        Result:=Result+diamond_calculate_function
      end
    end
  end
end
```

```
platinum_calculate_function:INTEGER
local
  p_position:INTEGER
do
  p_position:=0
  Result:=1
  orb_position:=orb_position+1
  from
    p_position:=1
  until
    p_position>2 or orb_position>orb_list.count
  loop
    if
      orb_list[orb_position]~"bronze"
    then
      Result:=Result+1
      orb_position:=orb_position+1
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"silver"
    then
      Result:=Result+2
      orb_position:=orb_position+1
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"gold"
    then
      Result:=Result+3
      orb_position:=orb_position+1
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"platinum"
    then
      Result:=Result+platinum_calculate_function
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"diamond"
    then
      Result:=Result+diamond_calculate_function
      p_position:=p_position+1
    end
  if
    p_position=3
  then
    Result:=Result*2
  end
  end
end
```

```
diamond_calculate_function:INTEGER
local
  p_position:INTEGER
do
  p_position:=0
  Result:=3
  orb_position:=orb_position+1
  from
    p_position:=1
  until
    p_position>3 or orb_position>orb_list.count
  loop
    if
      orb_list[orb_position]~"bronze"
    then
      Result:=Result+1
      orb_position:=orb_position+1
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"silver"
    then
      Result:=Result+2
      orb_position:=orb_position+1
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"gold"
    then
      Result:=Result+3
      orb_position:=orb_position+1
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"platinum"
    then
      Result:=Result+platinum_calculate_function
      p_position:=p_position+1
    elseif
      orb_list[orb_position]~"diamond"
    then
      Result:=Result+diamond_calculate_function
      p_position:=p_position+1
    end
  if
    p_position=4
  then
    Result:=Result*3
  end
  end
end
```

My idea for the scoring system is that every time an enemy is destroyed, his trophy name will be put into the trophy Array. (Interceptor: bronze orb; Grunt: silver orb; Fighter: gold orb; Carrier: platinum orb; Pylon: diamond orb). Then every round will settle the score of the trophy ARRAY and change the original score. First start the calculation from the trophy list. If it is bronze, silver, or gold trophies, it is just a simple addition. If it is platinum or diamond, it will enter the corresponding function. For example, platinum function. First of all, the first item in the platinum focus must be bronze. Then judge the second and third. The same way as calculate function. If the focus is full, the function result will times 2. But in the diamond function will times 3.

I will use these function and attribute in the score system

1.set_score(use to change score value)

2.orb_position(used to indicate which position in the trophy array is now)

3.calculate_function(use to calculate total score)

4.platinum_calculate_function(use to calculate platinum focus score)

5.diamond_calculate_function(use to calculate diamond focus score)

6.p_position(use to record which orb is now in platinum calculate function or diamond calculate function)

For example

The list of destroyed fighters is I,G,P,G,C,G,I,F,F,F.

The list of the orb will be B,S,D,S,P,S,B,G,G,G

So the count of the orb array is 10.

Firstly, we will go to the calculate_score function.

Because orb_list.count>10, so we will go to the loops.

1.orb_list[orb_position]~ “bronze”(orb_position=1) then Result=1(0+1) and orb_position=2(1+1)

Now in the calculate function.

2.orb_list[orb_position]~ “silver”(orb_position=2) then Result=3(1+2) and orb_position=3(2+1)

Now in the calculate function.

3.orb_list[orb_position]~ “diamond”(orb_position=3) then will go to diamond_calculate_function.

In the diamond function, the first one must is gold orb. So the Result=6(3+(3)), orb_position=4(3+1) and p_position=1(0+1)

Now in the diamond calculate function of calculate function.

4.orb_list[orb_position]~ “silver”(orb_position=4) then Result=8(3+(3+2)), orb_position=5(4+1) and p_position(d)=2(1+1)

Now in the diamond calculate function of calculate function

5.orb_list[orb_position]~ “platinum”(orb_position=5) then will go to platinum_calculate_function. In the diamond function, the first one must is bronze orb. So the Result=9(3+(3+2+(1))), orb_position=6(5+1) , p_position(d)=3(2+1) and p_position(p)=1(0+1)

Now in the platinum calculate function of diamond calculate function of calculate function

6.orb_list[orb_position]~ “silver”(orb_position=6) then Result=11(3+(3+2+(1+2)), orb_position=7(6+1), p_position(d)=3 and p_position(p)=2(1+1)

Now in the platinum calculate function of diamond calculate function of calculate function

7.orb_list[orb_position]~ “bronze”(orb_position=7) then Result=13(3+(3+2+(2*(1+2+1))), orb_position=8(7+1), p_position(d)=3 and p_position(p)=3(2+1)

Now in the diamond calculate function of calculate function

8.orb_list[orb_position]~ “gold”(orb_position=8) then Result=51(3+3*(3+2+8+3)), orb_position=9(8+1) and p_position(d)=4(3+1)

Now in the calculate function

9.orb_list[orb_position]~ “gold”(orb_position=9) then Result=54(51+3) and orb_position=10(9+1).

Now in the calculate function

10.orb_list[orb_position]~“gold”(orb_position=10) then Result=57(54+3) and orb_position=11(10+1).

Exit calculate function

• Information Hiding (what is hidden and may be changed? what is not hidden and stable?)

Yes, my scoring actions design and implementation satisfies the Information Hiding. The score calculation system can only be used in BOARD CLASS. So the calculation part of the score is hidden, but the score will change according to the trophy list. The trophy list is public, every enemy round, as long as they are destroyed, the trophy will be added to the trophy list.

• Single Choice Principle

Yes, my scoring actions design and implementation satisfies the Single Choice Principle. My scoring system only uses calculations based on the trophy list. If the code or type of this function changes, it will not affect other systems. In the same way, other systems will not affect the judgment of the scoring system.

• Cohesion

Yes, my scoring actions design and implementation is satisfies the Cohesion. Because the scoring system is only used to calculate the score, no other work will be performed

• Programming from the Interface, Not from the Implementation

No, my scoring actions design and implementation is not satisfies this. Since my system for calculating fractions only takes into account the score, there is no need to design an interface.