SE 3XA3: Module Interface Specification
TankWar

Team #212, Genius
Di Wu, 400117248, wud43
Jiahao Zhou, 400082351, zhouj56
Xinyu Huang, 400120376, huangx65

April 6, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
13 March 2020	1.0	Create the first version of the MIS
4 April 2020	2.0	Exchange some modules between Behaviour-Hiding Module and Software Decision Module. Make changes based on the modules we had improved after finishing the first version of MIS, which includes adding new modules, deleting modules, adding new functions along with its transition and exception, and deleting functions that are no longer needed in our project.

1 Module Hierarchy

Level 1	Level 2	Module Label
Hardware-Hiding Module		M1
Behaviour-Hiding Module	myTankControl Module	M2
	Screen Module	M3
	display Module	M4
	PvsE Module	M5
	PvsP Module	M6
	mapEditing Module	M7
	~~Map Module~~	
	main Module	M8
	highSpeedTank Module	M13
	doubleLifeTank Module	M14
	fastBulletTank Module	M15
	MapEditTank Module	M18
	enemyTank Module	M16
Software Decision Module	bullet Module	M9
	food Module	M10
	decTime Module	M11
	myTank Module	M12
	~~highSpeedTank Module~~	
	~~doubleLifeTank Module~~	
	~~fastBulletTank Module~~	
	~~enemyTank Module~~	
	~~positionType Module~~	
	Map Module	M17
	wall Module	M19

Table 2: Module Hierarchy

2 MIS of Bullet Module

2.1 Template Module

Bullet

2.2 Uses

pygame

2.3 Syntax

2.3.1 Exported Types

N/A

2.3.2 Exported Access Programs

Name	In	Out	Exceptions
new Bullet	-	-	-
setBulletSpeed	$\mathbb{Z}$	-	-
setBulletLife	$\mathbb{B}$	-	-
setBulletStrong	$\mathbb{B}$	-	-
changeImage	$\mathbb{Z}$,$\mathbb{Z}$	-	out of range
move	-	-	-

2.4 Semantics

2.4.1 State Variables

dir_x : $\mathbb{Z}$
dir_y : $\mathbb{Z}$
speed : $\mathbb{Z}$
life : $\mathbb{B}$
strong : $\mathbb{B}$
bullet : pygame loading image.

2.4.2 State Invariant

(-1 <= dir_x <= 1) & (-1 <= dir_x <= 1)

2.5 Environment Variables

bullet_up : pygame loading image.
bullet_down : pygame loading image.
bullet_left : pygame loading image.
bullet_right : pygame loading image.

2.5.1 Assumptions & Design Decisions

dir_x and dir_y are both integers from -1 to 1.

2.5.2 Access Routine Semantics

Bullet():

- transition:
  dir_x, dir _y, speed, life, strong, bullet are initialized to 0, 0, 6, False, False, bullet.up, and the information about the bullet's position will stored into rect.left and rect.right, which are 3 + 12 * 24, 3 + 24 * 24, respectively.

- excpetion:
  None.

setBulletSpeed(speed):

- transition:
  The speed of the bullet is changed to the input value.

- exception:
  None

setBulletLife(life):

- transition:
  The life of the bullet is changed to the input value.

- exception:
  None.

changeImage(dir_x, dir_y):

- transition:
  the direction of the bullet is changed to the input value, and bullet's images will be loaded continuously to show the moving in that direction.

- exception:
  (|dir_x| $>$ 1 or |dir_y| $>$ 1 will be a out of range exception)

move():

- transition:
  The position of the bullet will be changed continuously in a direction, and it will the life's value will be changed to False if the bullet hit something, such as a tank, wall or the boundary of the map.

- exception:
  None.

3 MIS of Decrease Time Module

3.1 Template Module

decTime

3.2 Uses

N/A

3.3 Syntax

3.3.1 Exported Types

N/A

3.3.2 Exported Access Programs

Name	In	Out	Exceptions
new decTime	$\mathbb{Z}$	-	-
subTime	-	$\mathbb{Z}$	-

3.4 Semantics

3.4.1 State Variables

sec : $\mathbb{Z}$
hour : $\mathbb{Z}$
minute : $\mathbb{Z}$

3.4.2 State Invariant

sec >= 0 & minute >= 0 & hour >= 0

3.4.3 Assumptions & Design Decisions

N/A

3.4.4 Access Routine Semantics

decTime(totalTime):

- transition:
  The value of sec is changed to the input totalTime, and the hour is set to sec/3600, then the sec is changed to sec% 3600, and the minute is set to sec/60, finally the value of sec is set to sec % 60.

- exception:
  None.

subTime():

- transition:
  If (sec $>$ 0), sec = sec - 1. If (sec == 0 and minute $>$ 0), minute -= 1 and sec is changed to 59. If (sec == 0 and hour $>$ 0), hour -= 1 and the sec is changed to 59. If sec. minute, and hour are all zeros, the game will end.

- exception:
  None.

4 MIS of DoubleLife Tank Module

4.1 Template Module

DoubleLifeTank

4.2 Uses

pygame, myTank, bullet, positionType

4.3 Inherit

MyTank

4.4 Syntax

4.4.1 Exported Types

N/A

4.4.2 Exported Access Programs

Name	In	Out	Exceptions
new DoubleLifeTank	CoordinateT	-	-
bulletproof_start	-	-	-
bulletproof_end	-	-	-
getBulletProof	-	$\mathbb{B}$	-

4.5 Semantics

4.5.1 State Variables

life : $\mathbb{Z}$
bulletProof : $\mathbb{B}$
ID : $\mathbb{Z}$
tank : pygame loading image
tank_R0 : subsurface of pygame loading image

tank_R1 : subsurface of pygame loading image
rect : rectangle object of pygame loading image
rect.left : $\mathbb{Z}$
rect.top ; $\mathbb{Z}$

4.5.2 State Invariant

N/A

4.5.3 Environement Variables

imagePath : path for a image

4.5.4 Assumptions & Design Decisions

- The double life tank class inherit my tank class. The double life tank has all the features of the my tank class, like level, move, shoot.
- The double life tanks can activate their ultimate skill, bullet proof. It can block the bullets for 3 seconds and this can be activated 3 times in a game.
- The double life tank has 6 life in a game.

4.5.5 Access Routine Semantics

doubleLifeTank(myTank.MyTank):

- transition:
  The tank's life, bulletProof, and ID are initialized to be 2, False, and 1, respectively. Load the image from imagePath to tank, and tank_R0, tank_R1 are set to be different parts of the image loaded, which represent the different conditions during moving. The rect used the method in pygame to get the tank_R0's position and the position information will are stored in rect.left and rect.top.
- exception:
  None.

bulletproof_start():

- transition:
  The value of bulletproof is set to be True.
- exception:
  None.

bulletproof_end():

- transition:
  The value of bulletproof is set to be False.

- exception:
  None.

getBulletProof():

- output:
  return the value of bulletproof.
- exception:
  None.

5 MIS of Enemy Tank Module

5.1 Template Module

EnemyTank

5.2 Uses

pygame, random, bullet

5.3 Syntax

5.3.1 Exported Types

N/A

5.3.2 Exported Access Programs

Name	In	Out	Exceptions
new EnemyTank	$\mathbb{Z}$, $\mathbb{Z}$, $\mathbb{B}$	-	-
shoot	-	-	$\|dir\_x\| > 1 \ \| \ \|dir\_y\| > 1$
move	set of Brick, set of Iron	-	-

5.4 Semantics

5.4.1 State Variables

flash : $\mathbb{B}$
times : $\mathbb{Z}$
kind : $\mathbb{Z}$
enemy_x_0 : pygame loading image
enemy_x_3 : pygame loading image
enemy_3_0 : pygame loading image

enemy_3_2 : pygame loading image
isred : $\mathbb{B}$
tank : pygame loading image
x : $\mathbb{Z}$
tank_R0 : subsurface of pygame loading image
tank_R1 : subsurface of pygame loading image
rect : rectangle object of pygame loading image
rect.left : $\mathbb{Z}$
rect.top : $\mathbb{Z}$
speed : $\mathbb{Z}$
dir_x : $\mathbb{Z}$
dir _y : $\mathbb{Z}$
life : $\mathbb{Z}$
bulletNotCooling : $\mathbb{B}$
bullet : Bullet()
dirChange : $\mathbb{B}$

5.4.2 State Invariant

N/A

5.4.3 Assumptions & Design Decisions

N/A

5.4.4 Access Routine Semantics

EnemyTank(x, kind, isred, y):

- transition:
  Flash and time is set to be False and 90, respectively. and the second input value is stored into the variable kind. If kind is not None, kind is set to be a integer randomly picked from [1,2,3,4]. Depending on the value kind, different images are stored into enemy_x_0 and enemy_x_3, and two images are stored into enemy_3_0, enemy_3_2 as well. If value of isred is not None, a random value from (True, False, False, False, False) is stored into it, and then if the value of isred is True, tank's value will set to enemy_x_3, otherwise it will be enemy_x_0. The first input value is assigned to the variable x, and if x is not None, a random value from [1,2,3] will be stored into x, and then (x = x -1). different parts of the tank image will be stored into tank_R0 and tank_R1, and the position information will be stored into rect.left and rect.top. The speed, dir_x, dir_y, bulletNotCooling, bullet, dirChange are set to be 1, 0, 1, 1, True, Bullet(), False, respectively. Finally, if kind equals to 2, the speed will be set as 3, and if kind is 3, the life will be set as 3.

- exception:
  None.

shoot():

- The life of the enemyTank's bullet is changed to True, and the image loaded depending on the direction of the bullet. The images are loaded continuously to represent to moving of the bullet in a direction.

- exception:
  (|dir_x| > 1) or (|dir_y| > 1)

move(tankGroup, brickGroup, ironGroup):

- transition:
  The rect value of the tank will be changed depending on the tank's direction and speed, and different subsurface of the tank image will be loaded tank_R1, and tank_R0 depending on the tank's moving direction. When the tank touches something, such as the other tanks, wall, or the boundary, a new random direction will be set.

- exception:
  None

6 MIS of Fast Bullet Tank Subclass Module

6.1 Template Module

FastBulletTank

6.2 Inherit

MyTank

6.3 Uses

Bullet, myTank, postitionType

6.4 Syntax

6.4.1 Exported Types

N/A

6.4.2 Exported Access Programs

Name	In	Out	Exceptions
new FastBulletTank	CoordinateT	-	-
doubleBullets	-	-	-

6.5 Semantics

6.5.1 State Variables

bullet: Bullet
bullet2: Bullet
ID : $\mathbb{Z}$
tank : pygame loading image
tank_R0 : subsurface of pygame loading image
tank_R1 : subsuface of pygame loading image
rect : rectangle object of pygame loading image
rect.left : $\mathbb{Z}$
rect.top : $\mathbb{Z}$

6.5.2 State Invariant

N/A

6.5.3 Assumptions & Design Decisions

- The fast bullet tank class inherit my tank class. The fast bullet tank has all the features of the my tank class, like level, move, shoot.
- The fast bullet tanks can activate their ultimate skill, double bullet. It can shoot double bullets 3 times.
- The fast bullet tanks have faster bullet than other tanks.

6.5.4 Access Routine Semantics

FastBulletTank(coordinate):

- transition:
  speed of bullet1 changed to 10, ID of bullet 1 is changed to 3. The image of the tank is stored into tank, and different parts of the subsurface are loaded to tank_R0 and tank_R1. rect.left and rect.top are changed to (3 + 24 * coordinate.x) and (3 + 24 * coordinate), respectively.
- exception: None.

doubleBullet():

- transition: life of bullet1 and bullet2 changed to True. According to the tank direction, changes the image(changeImage(int, int)) of bullet1 and bullet2. According to the tank level, changes the speed of the bullet1 and bullet2.
- exception: None.

7 MIS of Food Module

7.1 Template Module

food

7.2 Uses

pygame
random

7.3 Syntax

7.3.1 Exported Types

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
new Food	-	-	-
change	-	-	-

7.4 Semantics

7.4.1 State Variables

food_boom: pygame loading image
food_clock: pygame loading image
food_gun: pygame loading image
food_iron: pygame loading image
food_star: pygame loading image
image: pygame loading image
kind: $\mathbb{Z}$
life: $\mathbb{B}$
rect: rectangle object of pygame loading image
rect.left: $\mathbb{Z}$
rect.top: $\mathbb{Z}$

7.4.2 State Invariant

1 <= kind <= 5
100 <= rect.left <= 500
100 <= rect.top: <= 500

7.4.3 Assumptions & Design Decisions

- The food module is used to store the image of each food and random generate one of the food.
- The food object shall have 5 different kinds of buffs. The boom can destroy all enemy tank. The clock can pause all enemy tanks. The gun can make the bullets of the tank strong. The iron can add iron walls around the home. The star can upgrade the tank.
- The position of the food buffs will be random on the screen.
- The possibility of getting each food buff are equal.

7.4.4 Access Routine Semantics

Food():

- transition:
  The program should use pygame to load the image of the five food buff and store into the state variable food_boom, food_clock, food_gun, food_iron, food_star respectively. Then, the state variable kind will be assigned randomly among 1 to 5. The state variable image will store one of the image of the 5 food buffs determined by the state variable kind. Then, the state variable rect will be assigned with the rectangle object of the state variable image, and the position state variable rect.left and rect.top would be assigned with a random integer from 100 to 500. The state variable life would be set as False.
- exception: None.

change():

- transition:
  The state variable kind will be assigned randomly among 1 to 5. The state variable image will store one of the image of the 5 food buffs determined by the state variable kind. The position state variable rect.left and rect.top would be updated with a random integer from 100 to 500. The state variable life would be set as True.
- exception: None.

8 MIS of High Speed Tank Subclass Module

8.1 Template Module

highSpeedTank

8.2 Inherit

MyTank

8.3 Uses

pygame
bullet
positionType

8.4 Syntax

8.4.1 Exported Types

N/A

8.4.2 Exported Access Programs

Name	In	Out	Exceptions
new highSpeedTank	CoordinateT	-	-
leap_start	-	-	-
leap_end	-	-	-

8.5 Semantics

8.5.1 State Variables

speed: $\mathbb{Z}$
ID: $\mathbb{Z}$
tank: pygame loading image
tank_R0: subsurface of pygame loading image
tank_R1: subsurface of pygame loading image
rect: rectangle object of pygame loading image
rect.left: $\mathbb{Z}$
rect.top: $\mathbb{Z}$

8.5.2 State Invariant

ID = 2

8.5.3 Assumptions & Design Decisions

- The high speed tank class is inherited from myTank class, so it has all the operations and features of the super call.
- The high speed tank would have a ultimate skill called leap, which can boost the speed of the tank.

8.5.4 Access Routine Semantics

highSpeedTank(coordinate):

- transition:
  The program should set the state variable ID as 2. Then, it should use pygame to load the image of the high speed tank and store into the state variable tank.The state variable tank_R0, and tank_R1 will use the subsurface of the state variable to abstract two moving stage images. Then, the state variable rect will be assigned with the rectangle object of the state variable tank, and the position state variable rect.left and rect.top would be assigned by the coordinate.x and coordinate.y respectively.
- exception: None.

leap_start():

- transition:
  The function would increase the state variable speed by 3.
- exception: None.

leap_end():

- transition:
  The function would set the speed to original speed.
- exception: None.

9 MIS of Map Module

9.1 Template Module

Map

9.2 Uses

pygame wall postionType typing

9.3 Syntax

9.3.1 Exported Types

CoordinateT: NameTuple of {x: int, y: int}

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
new Map	-	-	-
addBrick	CoordinateT	-	-
addIron	CoordinateT	-	-
addHome	CoordinateT	-	-
loadBrickIron	string	-	-
saveMap	string	-	-
new PVEMap	-	-	-
loadPVEMap	string	-	-
new PVPMap	-	-	-
loadPVPMap	string	-	-
CoordinateT	NameTuple	-	-

9.4 Semantics

9.4.1 State Variables

brickGroup: pygame sprite group
ironGroup: pygame sprite group
homeGroup: pygame sprite group

9.4.2 State Invariant

None

9.4.3 Assumptions & Design Decisions

- The Map class is designed as the super class which contains all date about the position of the bricks, irons, homes in the map.
- The PVEMap is designed as the subclass of the Map class.
- The PVPMap is designed as the subclass of the Map class.

9.4.4 Access Routine Semantics

Map():

- transition:
  The function would initialize the Map object. The state variables brickGroup, ironGroup, homeGroup would be assigned by an empty pygame sprite group.
- exception: None.

addBrick(coordinate):

- transition:
  The function would create a Brick() object. Them, it uses coordinate.x and coordinate.y to determine the position of the brick on the screen. After the brick setting up, the Brick object will be added into the state variable brickGroup.

- exception: None.

addIron(coordinate):

- transition:
  The function would create a Iron() object. Them, it uses coordinate.x and coordinate.y to determine the position of the iron on the screen. After the iron setting up, the Iron object will be added into the state variable ironGroup.

- exception: None.

addHome(coordinate):

- transition:
  The function would create a Home() object. Them, it uses coordinate.x and coordinate.y to determine the position of the home on the screen. After the home setting up, the Home object will be added into the state variable homeGroup.

- exception: None.

loadBrickIron(path):

- transition:
  The function would read the map storing file according to the path. It can read the format of the map storing file as the next function saveMap() generated. According to the coordinates stored in the file, it should create bricks as well as irons object, and add them into the state variable brickGroup and homeGroup.

- exception: None.

saveMap(path):

- transition:
  The function should save all the coordinates of the bricks and irons in the state variable brickGroup and homeGroup into the following format.

  Brick:
  4 2
  4 3
  Iron:
  0 2
  0 3
  Number of Brick:
  2

Number of Iron:
2

First row is "Brick:". Each following row represents a coordinate of a brick in the map. The left integer is x coordinate and the right integer is y coordinate in a row. After all bricks in the state variable brickGroup is written into the map, the iron would follow the same format start with 'Iron:' in the next row. Then, the coordinates of the iron in the ironGroup is written into the file. After that, the 'Number of Brick:' is written into the file and the next row is the number of the bricks in the brickGroup in the map. Then, the 'Number of Iron:' is written into the file and the next row is the number of the irons in the ironGroup in the map.

- exception: None.

PVEMap():

- transition:
  PVEMap class is inherited from Map class. The function would initialize the PVEMap object. The state variables brickGroup, ironGroup, homeGroup would be assigned by an empty pygame sprite group.

- exception: None.

loadPVEMap(path):

- transition:
  The function would read the map storing file according to the path. It can read the format of the map storing file as the function saveMap() generated. According to the coordinates stored in the file, it should create bricks as well as irons object, and add them into the state variable brickGroup and homeGroup. Moreover, it should also generate Home object with CoordinateT(12,24) to place the home in the middle of the bottom of the map and add the home into the state variable homeGroup.

- exception: None.

PVPMap():

- transition:
  PVPMap class is inherited from Map class. The function would initialize the PVPMap object. The state variables brickGroup, ironGroup, homeGroup would be assigned by an empty pygame sprite group.

- exception: None.

loadPVPMap(path):

- transition:
  The function would read the map storing file according to the path. It can read the format of the map storing file as the function saveMap() generated. According to the

coordinates stored in the file, it should create bricks as well as irons object, and add them into the state variable brickGroup and homeGroup. Moreover, it should also generate 2 Home objects with CoordinateT(0,12) as well as CoordinateT(24,12) to place the homes in the middle of the left and right boundary of the map and add the home into the state variable homeGroup.

- exception: None.

CoordinateT(coordinate):

- transition:
  The function would create a position format which saves the coordinate based on the input.
- exception: None.

10 MIS of My Tank Superclass Module

10.1 Template Module

MyTank

10.2 Uses

Pygame bullet positionType

10.3 Syntax

10.3.1 Exported Types

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
New myTank	CoordinateT	-	-
shoot	-	-	-
levelUp	-	-	-
levelDown	-	-	-
moveUp	set of Tank, set of Brick, set of Iron	$\mathbb{B}$	-
moveDown	set of Tank, set of Brick, set of Iron	$\mathbb{B}$	-
moveLeft	set of Tank, set of Brick, set of Iron	$\mathbb{B}$	-
moveRight	set of Tank, set of Brick, set of Iron	$\mathbb{B}$	-
setSpeed	$\mathbb{Z}$	-	-

10.4 Semantics

10.4.1 State Variables

life: $\mathbb{Z}$
level: $\mathbb{Z}$
speed: $\mathbb{Z}$
bullet: Bullet object
bulletNotCooling: $\mathbb{B}$
tank: pygame loading image
tank_R0: subsurface of pygame loading image
tank_R1: subsurface of pygame loading image
rect: rectangle object of pygame loading image
rect.left: $\mathbb{Z}$
rect.top: $\mathbb{Z}$
dir_x: $\mathbb{Z}$
dir_y: $\mathbb{Z}$

10.4.2 State Invariant

level = $level \in \mathbb{Z} | 0 \leq level \leq 2$
dir_x = $level \in \mathbb{Z} | -1 \leq dir\_x \leq 1$
dir_y = $level \in \mathbb{Z} | -1 \leq dir\_x \leq 1$

10.4.3 Environmental Variable

Tank.png: png picture. The Tank.png is the picture containing the image of tanks moving towards up, down, left, right in two phases moving or stop respectively.

10.4.4 Assumptions & Design Decisions

- MyTank is a super class which represents the tank object.
- A MyTank object should have the methods to move towards up, down, left, right, to level up as well as down, and to shoot bullets as the basic functionalities of a tank.

10.4.5 Access Routine Semantics

MyTank():

- transition:
  The function would initialize the MyTank object. It should initialise the state variable life and level to 1. It should initialise the state variable speed to 3. It should initialise the state variable bullet with a Bullet() object and bulletNotCooling with True. It should use pygame to load the image 'Tank.png' and store into the state variable

tank.The state variable tank_R0, and tank_R1 will use the subsurface of the state variable to abstract two moving stage images. Then, the state variable rect will be assigned with the rectangle object of the state variable tank, and the position state variable rect.left and rect.top would be assigned by the coordinate.x and coordinate.y respectively. The state variable dir_x and dir_y are initialized by assigning 0 and -1 respectively.

- exception: None.

shoot():

- transition:
  The function would set the state variable bullet life to be true by bullet.setBulletLife(True) and change the direction of image of the bullet by changeImage(self.dir_x, self.dir_y). While, the function also change the position of the bullet according to the shooting direction. Based on the level of the tank, the bullet speed and strong variable should be set to the corresponding value. Level 1 tank would have speed 16 bullet. Level 1 tank would have speed 16 bullet. Level 2 tank would have speed 16 bullet with bullet strong property True.

- exception: None.

levelUp():

- transition:
  If the level is less than 2, the level of the tank will increase by 1.

- exception: None.

levelDown():

- transition:
  If the level is greater than 0, the level of the tank will decrease by 1. If the level is 0, the level is remaining unchanged.

- exception: None.

moveUp(tankGroup, brickGroup, ironGroup):

- output:
  The tank object shall move toward up by using state variable rect.move() function to move the object. If there are other objects on the way up like brick, iron, other tanks in the brickGroup, ironGroup, tankGroup, the tank will be blocked by these objects. If the tank is moving without blocking, the function will return True. If the tank is moving with blocking, the function will return False.

- exception: None.

moveDown(tankGroup, brickGroup, ironGroup):

- output:
  The tank object shall move toward down by using state variable rect.move() function to move the object. If there are other objects on the way down like brick, iron, other tanks in the brickGroup, ironGroup, tankGroup, the tank will be blocked by these objects. If the tank is moving without blocking, the function will return True. If the tank is moving with blocking, the function will return False.

- exception: None.

moveLeft(tankGroup, brickGroup, ironGroup):

- output:
  The tank object shall move toward left by using state variable rect.move() function to move the object. If there are other objects on the way left like brick, iron, other tanks in the brickGroup, ironGroup, tankGroup, the tank will be blocked by these objects. If the tank is moving without blocking, the function will return True. If the tank is moving with blocking, the function will return False.

- exception: None.

moveRight(tankGroup, brickGroup, ironGroup):

- output:
  The tank object shall move toward right by using state variable rect.move() function to move the object. If there are other objects on the way right like brick, iron, other tanks in the brickGroup, ironGroup, tankGroup, the tank will be blocked by these objects. If the tank is moving without blocking, the function will return True. If the tank is moving with blocking, the function will return False.

- exception: None.

setSpeed(speed):

- transition:
  The state variable speed is assigned by the input speed.

- exception: None.

11 MIS of My Tank Control Module

11.1 Template Module

myTankControl

11.2 Uses

Pygame myTank

11.3 Syntax

11.3.1 Exported Types

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
operatePlayer1	pygame loading audio, pygame key press, $\mathbb{Z}$, $\mathbb{Z}$, MyTank, pygame sprite group of MyTanks, pygame sprite group of Bricks, pygame sprite group of Irons, $\mathbb{B}$	-	-
operatePlayer2	pygame loading audio, pygame key press, $\mathbb{Z}$, $\mathbb{Z}$, MyTank, pygame sprite group of MyTanks, pygame sprite group of Bricks, pygame sprite group of Irons, $\mathbb{B}$	-	-
operatePlayerME	pygame key press, moving movdir1, MyTank, pygame sprite group of Tanks, bgMap, running_T1	-	-

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 State Invariant

N/A

11.4.3 Assumptions & Design Decisions

- myTankControl would handle the key press from the two players respectively and control the tank to move, shoot, activate ultimate skills of the tanks.

11.4.4 Access Routine Semantics

operatePlayer1(fire_sound, key_pressed, moving, movdir1, myTank, allTankGroup, brickGroup, ironGroup, running_T1):

- transition:
  The function shall identify the key press from the users' keyboard. If the key 'w', 's', 'a', 'd' are pressed, the tank should move up, down, left, right respectively. If the key 'j' and 'k' are pressed, the tank will shoot bullets and activate the ultimate skill.

- exception: None.

operatePlayer2(fire_sound, key_pressed, moving2, movdir2, myTank, allTankGroup, brickGroup, ironGroup, running_T2):

- transition:
  The function shall identify the key press from the users' keyboard. If the arrow key 'up', 'down', 'left', 'right' are pressed, the tank should move up, down, left, right respectively. If the key ',' and '.' are pressed, the tank will shoot bullets and activate the ultimate skill.

- exception: None.

operatePlayerME(key_pressed, moving, movdir1, myTank, allTankGroup, bgMap, running_T1):

- transition:
  The function shall identify the key press from the users' keyboard. If the key 'w', 's', 'a', 'd' are pressed, the tank should move up, down, left, right respectively. If the key 'j' is pressed, the tank will shoot bullets.

- exception: None.

12 MIS of Wall Module

12.1 Template Module

Wall

12.2 inherit

pygame.sprite.Sprite

12.3 Uses

Pygame

12.4 Syntax

12.4.1 Exported Types

N/A

12.4.2 Exported Access Programs

Name	In	Out	Exceptions
new Brick	-	-	-
new Iron	-	-	-
new Home	$\mathbb{Z}$	-	-

12.5 Semantics

12.5.1 State Variables

Brick Class:
image: pygame loading image
rect: rectangle object of pygame loading image
Iron Class:
image: pygame loading image
rect: rectangle object of pygame loading image
Home Class:
image: pygame loading image
rect: rectangle object of pygame loading image
homeID: $\mathbb{Z}$

12.5.2 State Invariant

None

12.5.3 Environmental variable

brickImage: brick.png, image of brick ironImage: iron.png, image of iron homeImage: home.png, image of home

12.5.4 Assumptions & Design Decisions

- Wall module contains 3 classes, including Brick, Iron, and Home.
- All the 3 classes in the module consists the map of the game.

12.5.5 Access Routine Semantics

Brick():

- transition:
  The function create a Brick object and load the brickImage into the state variable image. The state variable rect is assigned by the rectangle object of the state variable image.get_rect().
- exception: None.

Iron():

- transition:
  The function create a Iron object and load the ironImage into the state variable image. The state variable rect is assigned by the rectangle object of the state variable image.get_rect().

- exception: None.

Home():

- transition:
  The function create a Home object and load the homeImage into the state variable image. The state variable rect is assigned by the rectangle object of the state variable image.get_rect().
- exception: None.

13 ~~MIS of positionType Module~~

13.1 ~~Template Module~~

~~positionType~~

13.2 ~~Inherit~~

~~NamedTuple~~

13.3 ~~Uses~~

~~typing~~

13.4 ~~Syntax~~

13.4.1 ~~Exported Types~~

~~CoordinateT = tuple of (x: int, y: int)~~

13.4.2 ~~Exported Access Programs~~

~~N/A~~

13.5 ~~Semantics~~

13.5.1 ~~State Variables~~

~~N/A~~

13.5.2 ~~State Invariant~~

~~N/A~~

13.5.3 ~~Assumptions & Design Decisions~~

~~N/A~~

13.5.4 ~~Access Routine Semantics~~

~~N/A~~

14 MIS of MapEditTank Module

14.1 Template Module

MapEditTank

14.2 Uses

pygame, bullet, Map

14.3 Syntax

14.3.1 Exported Types

N/A

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
new MapEditTank	coordinate	-	-
shoot	-	-	-
moveUp	tankGroup, brickGroup, ironGroup	boolean	-
moveDown	tankGroup, brickGroup, ironGroup	boolean	-
moveLeft	tankGroup, brickGroup, ironGroup	boolean	-
moveRight	tankGroup, brickGroup, ironGroup	boolean	-

14.4 Semantics

14.4.1 State Variables

N/A

14.4.2 State Invariant

N/A

14.4.3 Assumptions & Design Decisions

N/A

14.4.4 Access Routine Semantics

MapEditTank(coordinate):

- transition:
  The function create a MapEditTank object and set the input as the coordinate location of the tank.

- exception:
  None.

shoot():

- transition:
  The function allows MapEditTank to shoot by setting the bullet life to True. It will also controls the movement of the bullet.

- exception:
  None.

moveUp(tankGroup, brickGroup, ironGroup):

- transition:
  The tank object shall move toward up by using state variable rect.move() function tomove the object. If there are other objects on the way up like brick, iron in the brickGroup, ironGroup, the tank will not be blocked by these objects since this kind of tank will only be created in map editing mode.

- exception:
  None.

moveDown(tankGroup, brickGroup, ironGroup):

- transition:
  The tank object shall move down by using state variable rect.move() function to move the object. If there are other objects on the way up like brick, iron in the brickGroup, ironGroup, the tank will not be blocked by these objects since this kind of tank will only be created in map editing mode.

- exception:
  None.

moveLeft(tankGroup, brickGroup, ironGroup):

- transition:
  The tank object shall move left by using state variable rect.move() function to move the object. If there are other objects on the way up like brick, iron in the brickGroup, ironGroup, the tank will not be blocked by these objects since this kind of tank will only be created in map editing mode.

- exception:
  None.

moveRight(tankGroup, brickGroup, ironGroup):

- transition:
  The tank object shall move right by using state variable rect.move() function to move the object. If there are other objects on the way up like brick, iron in the brickGroup, ironGroup, the tank will not be blocked by these objects since this kind of tank will only be created in map editing mode.

- exception:
  None.

15 MIS of Display Module

15.1 Template Module

display

15.2 Uses

pygame, wall

15.3 Syntax

15.3.1 Exported Types

N/A

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawPVE	homeSurvive mytankGroup, deadCount1 deadCount2 switch_R1_R2_image enemyCouldMove enemyNumber prop moving running_T1 running_T2 bgMap background_image screen delay myTank_T1 myTank_T2 allEnemyGroup allTankGroup appearance enemyBulletGroup redEnemyGroup greenEnemyGroup otherEnemyGroup	homeSurvive mytankGroup, deadCount1 deadCount2 switch_R1_R2_image running_T1 running_T2 allTankGroup enemyNumber myTank_T1.bullet.life myTank_T1.bullet.rect.left myTank_T1.bullet.rect.right myTank_T2.bullet.life myTank_T2.bullet.rect.left myTank_T2.bullet.rect.right moving myTank_T1.rect.left myTank_T1.rect.top myTank_T1.level myTank_T2.rect.left myTank_T2.rect.top prop.life enemyCouldMove	-
drawPVP	homeDead mytankGroup deadCount1 deadCount2 switch_R1_R2_image moving running_T1 running_T2 bgMap background_image screen delay myTank_T1 myTank_T2 allTankGroup	homeDead mytankGroup deadCount1 deadCount2 switch_R1_R2_image running_T1 running_T2 allTankGroup myTank_T1.bullet.life myTank_T1.bullet.rect.left myTank_T1.bullet.rect.right myTank_T2.bullet.life myTank_T2.bullet.rect.left myTank_T2.bullet.rect.right moving myTank_T1.rect.left myTank_T1.rect.top myTank_T1.level myTank_T2.rect.left myTank_T2.rect.top	-

Name	In	Out	Exceptions
drawME	bgMap deadCount1 mytankGroup switch_R1_R2_image moving running_T1 background_image screen delay myTank_T1 allTankGroup	bgMap allTankGroup mytankGroup switch_R1_R2_image moving running_T1 myTank_T1.bullet.life myTank_T1.bullet.rect.left myTank_T1.bullet.rect.right myTank_T1.rect.left myTank_T1.rect.top	-

15.4 Semantics

15.4.1 State Variables

N/A

15.4.2 State Invariant

N/A

15.5 Environment Variables

bang.sound : A audio file with type .wav.

15.5.1 Assumptions & Design Decisions

- Display.py is designed to be used to draw every elements on the gaming screen.
- It can also determine the changes in the game and show the results of changes on the screen.

15.5.2 Access Routine Semantics

drawPVE(homeSurvive, mytankGroup, deadCount1, deadCount2, switch_R1_R2_image, enemyCouldMove, enemyNumber, prop, moving, running_T1, running_T2, bgMap, background_image, screen,delay, myTank_T1, myTank_T2, allEnemyGroup, allTankGroup, appearance, enemyBulletGroup, redEnemyGroup, greenEnemyGroup, otherEnemyGroup):

- transition:
  The drawPVE function is used in PVE module. After calling this function, all the functions related to PVE mode will be used and build a complete PVE environment together. The function returns homeSurvive, mytankGroup, deadCount1, deadCount2, switch_R1_R2_image, running_T1, running_T2, allTankGroup, enemyNumber,

myTank_T1.bullet.life, myTank_T1.bullet.rect.left,
myTank_T1.bullet.rect.right, myTank_T2.bullet.life,
myTank_T2.bullet.rect.left, myTank_T2.bullet.rect.right, moving, myTank_T1.rect.left,
myTank_T1.rect.top, myTank_T1.level, myTank_T2.rect.left, myTank_T2.rect.top,
prop.life, enemyCouldMove.

- exception:
  None.

drawPVP(homeDead, mytankGroup, deadCount1, deadCount2, switch_R1_R2_image, moving, running_T1, running_T2, bgMap, background_image, screen,delay, myTank_T1, myTank_T2, allTankGroup):

- transition:
  The drawPVP function is used in PVP module. After calling this function, all the functions related to PVP mode will be used and build a complete PVP environment together. The function returns homeDead, mytankGroup, deadCount1, deadCount2, switch_R1_R2_image, running_T1, running_T2, allTankGroup, myTank_T1.bullet.life, myTank_T1.bullet.rect.left, myTank_T1.bullet.rect.right, myTank_T2.bullet.life, myTank_T2.bullet.rect.left, myTank_T2.bullet.rect.right, moving, myTank_T1.rect.left, myTank_T1.rect.top, myTank_T1.level, myTank_T2.rect.left, myTank_T2.rect.top.

- exception:
  None.

drawME(bgMap, deadCount1, mytankGroup, switch_R1_R2_image, moving, running_T1, background_image, screen, delay, myTank_T1, allTankGroup):

- transition:
  The drawME function is used in map editing module. After calling this function, all the functions related to map editing mode will be used and build a complete map editing environment together. The function returns bgMap, allTankGroup, mytankGroup, switch_R1_R2_image, moving, running_T1, myTank_T1.bullet.life, myTank_T1.bullet.rect.left, myTank_T1.bullet.rect.right, myTank_T1.rect.left, myTank_T1.rect.top.

- exception:
  None.

15.5.3 Local Functions

checkCollideME(bullet, bgMap):

- transition:
  The bullet represents the bullet object shot by the tank. The bgMap represents the map object. checkCollideME function is used to determine the results for every colliding cases in the map editing mode, including collides between bullets and brick walls, bullets and iron walls. The function will return bullet and bgMap so that the game system will get the changes of these two variables.

- exception:
  None

checkCollidePVP(bullet, bgMap, myTank, homeDead):

- transition:
  The bullet represents the bullet object shot by the tank. The bgMap represents the map object. The myTank object represents the tank which will be shot in the PVP mode. The homeDead is a variable which is used to record whether the home base is destroyed. checkCollidePVP function is used to determine the results for every colliding cases in the game, including collides between bullets and different kinds of tanks, bullets and brick walls, bullets and iron walls. The function will return bullet, bgMap, myTank, homeDead so that the game system will get the changes of these four variables.

- exception:
  None.

checkCollidePVE(enemyBulletGroup, bullet, redEnemyGroup, greenEnemyGroup, otherEnemyGroup, bgMap, prop, enemyNumber, homeSurvive):

- transition:
  The enemyBulletGroup reprensents the group of enemy tanks generated in this game. The bullet represents the bullet object shot by the tank. The bgMap represents the map object. The redEnemyGroup, greenEnemyGroup, and otherEnemyGroup represents different kinds of enemy tanks. The prop is used to generate food in the PVE game. The enemyNumber is used to control the total enemy number on the PVP battlefield. The homeSurvive is a variable to determine whether the home base still exists. CheckCollidePVE function is used to determine the results for every colliding cases in the game, including collides between bullets and different kinds of enemies, bullets and brick walls, bullets and iron walls. The function will return enemyBulletGroup, bullet, redEnemyGroup, greenEnemyGroup, otherEnemyGroup, bgMap, prop, enemyNumber, homeSurvive so that the game system will get the changes of these four variables.

- exception:
  None.

drawBG(background_image, screen):

- transition:
  Generates the background image.

- exception:
  None.

drawBrick(bgMap, screen):

- transition:
  Display every unit of brick walls from the Brick group of the bgMap.

- exception:
  None.

drawIron(bgMap, screen):

- transition:
  Display every unit of iron walls from the iron group of the bgMap.

- exception:
  None.

drawHome(bgMap, screen):

- transition:
  Display all the home base from the home group of the bgMap.

- exception:
  None.

drawTank_1(deadCount1, switch_R1_R2_image, running_T1, delay, myTank_T1, screen):

- transition:
  The drawTank_1 function will display the player 1's corresponding tank image on the screen. It will also change the image in order to make the tank look like it is moving based on the value of switch_R1_R2_image. The function returns switch_R1_R2_image, running_T1 to update the latest variables.

- exception:
  None.

drawTank_2(deadCount2, switch_R1_R2_image, running_T2, myTank_T2, screen):

- transition:
  The drawTank_1 function will display the player 2's corresponding tank image on the screen. It will also change the image in order to make the wheels look like it is moving based on the value of switch_R1_R2_image. The function returns running_T2 to update the latest variables.

- exception:
  None.

drawEnemyTank(switch_R1_R2_image, enemyCouldMove, bgMap, allEnemyGroup, screen, allTankGroup, appearance):

- transition:
  The drawEnemyTank function will display the images of enemy tanks on the screen. Dirrerent kinds of enemy tanks will generate different images. It will also change the image in order to make the wheels look like moving based on the value of switch_R1_R2_image. The system returns switch_R1_R2_image and allTankGroup.

- exception:
  None.

drawMyBullet(deadCount, homeSurvive, enemyNumber, prop, bgMap, myTank, screen, enemyBulletGroup, redEnemyGroup, greenEnemyGroup, otherEnemyGroup):

- transition:
  The drawMyBullet function will display the bullets if the player shoots by pressing the corresponding key and make them disappear based on the collide check. The function returns deadCount, homeSurvive, enemyNumber, myTank.bullet.life, myTank.bullet.rect.left, myTank.bullet.rect.right.

- exception:
  None.

drawEnemyBullet(homeSurvive, mytankGroup, deadCount1, deadCount2, enemyCouldMove, moving, bgMap, allEnemyGroup, enemyBulletGroup, screen, myTank_T1, myTank_T2):

- transition:
  The drawEnemyBullet function will display the bullets if the enemy shoots and make them disappear in different situations using enemy tanks' bullet collide check. The function returns homeSurvive, mytankGroup, deadCount1, deadCount2, moving, myTank_T1.rect.left, myTank_T1.rect.top, myTank_T1.level, myTank_T2.rect.left, myTank_T2.rect.top.

- exception:
  None.

drawFood(enemyNumber, enemyCouldMove, prop, bgMap, screen, allEnemyGroup, myTank):

- transition:
  The drawFood function will display the food if a food is generated. Also the function will determine what kinds of effects will be generated if the tank eats a food. The function returns prop.life, enemyCouldMove, enemyNumber.

- exception:
  None.

drawMyBulletPVP(deadCount1, deadCount2, myTank_T2, homeDead, bgMap, myTank_T1, screen):

- transition:
  The drawMyBulletPVP function will display the bullets in PVP mode if the player shoots by pressing the corresponding key and make them disappear based on the collide check. The function returns deadCount2, myTank_T2.rect.left, myTank_T2.rect.top, myTank_T2.level,
  homeDead, myTank_T1.bullet.life, myTank_T1.bullet.rect.left, myTank_T1.bullet.rect.right.

- exception:
  None.

drawBulletME(bgMap, myTank_T1, screen):

- transition:
  The drawMyBulletME function will display the bullets in map editing mode if the player shoots by pressing the corresponding key and make them disappear based on the collide check. The function returns bgMap, myTank_T1, and screen.

- exception:
  None.

16 MIS of Screen Module

16.1 Template Module

screen

16.2 Uses

pygame, MyTank, positionType, os, highSpeedTank, doubleLifeTank, fastBulletTank.

16.3 Syntax

16.3.1 Exported Types

N/A

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
StartGame	-	-	-
Menue	-	-	-
chooseTankScreen	CoordinateT, CoordinateT	MyTank, MyTank	-
loadingMapScreen	File	Map	-
operationInstructPlay	-	-	-
operationInstructMap	-	-	-
ruleScreen	Rule	-	-
endScreen_PVE	$\mathbb{Z}$	-	-
endScreen_PVP	$\mathbb{Z}$	-	-
saveScreenME	File	Map	-
chooseMapME	-	selection	-

16.4 Semantics

16.4.1 State Variables

clock : pygame clock
resolution = $\mathbb{Z}$, $\mathbb{Z}$
screen : pygame display set mode
chooseTank :pygame loading image
c1 : CoordinateT
c2 : CoordinateT
myTank_T1 : MyTank
myTank_T2 : MyTank
keypress : pygame key get pressed
image : pygame loading image
my_font : pygame font
coor : $\mathbb{Z}$
countt : $\mathbb{Z}$
confirm : $\mathbb{B}$
PVERule : pygame loading image
PVPRule : pygame loading image
rule : $\mathbb{Z}$
PVEWin : pygame loading image
PVELose : pygame loading image
PVPWin1 : pygame loading image
PVPWin2 : pygame loading image

PVPDraw : pygame loading image

16.4.2 State Invariant

N/A

16.4.3 Assumptions & Design Decisions

N/A

16.4.4 Access Routine Semantics

StartGame():

- transition:
  Get the starting screen image and show on the screen.
- output:
  None
- exception:
  None.

Menue():

- transition:
  Get the menu screen image and show on the screen.
- output:
  None
- exception:
  None.

chooseTankScreen(coordinate_T1, coordinate_T2):

- transition:
  clock and screen are created as pygame clock, and set display mode, respectively. resolution, c1, c2 are initialized to be (630, 630), coordinate_T1, coordinate_T2, respectively and myTank_T1, myTank_T2 are initialized to doubleLifeTank(c1), highSpeedTank(c2), respectively. keypress is set to pygame.key.get_pressed(), and the value of confirm is set to False. When the players start to choose the tanks, different tank type will be assigned to myTank_T1, and myTank_T2. The keys 1, 2, 3 stand for doubleLifeTank, highSpeedTank, fastBulletTank for myTank_T1, respectively.The keys 7, 8, 9 are doubleLifeTank, highSpeedTank, fastBulletTankare for myTank_T2.
- output:
  myTank_T1, myTank_T2 are returned.

- exception:
  None.

loadingMapScreen(File):

- transition:
  clock and screen are created as pygame clock, and set display mode, respectively. resolution are initialized to (630, 630). My_font is changed to pygame.font.Font(None,40), coor and countt are changed to 241 and 1. respectively. For i in the File, string = str(countt) + ". " + str(i) , coor and countt are changed to coor + 30 and countt + 1, respectively. The value of confirm is changed to False. returnMap = File[0], keypress = pygame.key.get_pressed(). Different values are assigned to returnMap depending on the key pressed by the players.

- output:
  returnMap is returned.

- exception:
  None.

operationInstructPlay():

- transition:
  Get the operation instruction screen image and show on the screen.

- output:
  None

- exception:
  None.

operationInstructMap():

- transition:
  Get the operation instruction screen image for map editing and show on the screen.

- output:
  None

- exception:
  None.

ruleScreen(Rule):

- transition:
  clock and screen are created as pygame clock, and set display mode, respectively. resolution are initialized to (630, 630). The image are loaded to PVERule and PVPRule, and if rule equals to 0, PVERule will be assigned to image, otherwise PVPRule will be assigned to image. The value of confirm is changed to False, keypress = pygame.key.get_pressed(). If the players press the key Return, confirm will be set to True.

- exception:
  None

endScreen_PVE():

- transition:
  clock and screen are created as pygame clock, and set display mode, respectively. resolution are initialized to (630, 630). The images are loaded to PVEWin and PVPLose. If result is True, PVEWin will be assigned to image, otherwise PVELose will be assigned. The value of confirm is changed to False, keypress = pygame.key.get_pressed(). If the players press the key Return, confirm will be set to True.

- exception:
  None.

endScreen_PVP():

- transition:
  clock and screen are created as pygame clock, and set display mode, respectively. resolution are initialized to (630, 630). The images are loaded to PVPWin1, PVPWin2, and PVPDraw. If result equals to 1, PVPWin1 will be assigned to image, and if result equals to 2, PVPWin2 will be assigned to image, otherwise PVPDraw will be assigned to image. The value of confirm is changed to False, keypress = pygame.key.get_pressed().If the players press the key Return, confirm will be set to True.

- exception:
  None.

saveScreenME():

- transition:
  Get the save map screen image for map editing and show on the screen.

- output:
  Map is returned.

- exception:
  None.

chooseMapME():

- transition:
  Get the choose map screen image for map editing and show at the beginning of map editing mode.

- output:
  selection between PVP map and PVE map is returned.

- exception:
  None.

17 MIS of P vs E Module

17.1 Template Module

PvsE

17.2 Uses

pygame
Bullet
BulletControl
MyTankControl
DoubleLifeTank
EnemyTank
FastBulletTank
Food
HighSpeedTank
Map

17.3 Syntax

17.3.1 Exported Types

N/A

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
PvsE	-	-	-

17.4 Semantics

17.4.1 State Variables

game_result : $\mathbb{Z}$
coordinate_T1 : CoordinateT
coordinate_T2 : CoordinateT
myTank_T1 : MyTank
myTank_T2 : MyTank
returnMap : Map
resolution ; $\mathbb{Z}$, $\mathbb{Z}$
screen : pygame display mode
background_image : pygame loading image
home_image : pygame loading image
home_destroyed_image : pygame loading image

bang_sound : pygame loading sound
fire_sound : pygame loading sound
start_sound : pygame loading sound
allTankGroup : pygame sprite group
myTankGroup : pygame sprite group
bgMap : PVEMap
prop : Food
deadCount1 : $\mathbb{Z}$
deadCount2 : $\mathbb{Z}$
strtime1 : String
strtime2 : String
deadline : datetime
now : datetime
subtime : $\mathbb{Z}$
ttime = decTime
MYBULLETNOTCOOLINGEVENT : pygame event
BULLETP : pygame event
LEAP : pygame event
Time : pygame event
delay : $\mathbb{Z}$
moving : $\mathbb{Z}$
movdir : $\mathbb{Z}$
moving2 : $\mathbb{Z}$
movdir2 : $\mathbb{Z}$
enemyNumber : $\mathbb{Z}$
enemyCouldMove : $\mathbb{B}$
switch_R1_R2_image : $\mathbb{B}$
homeDead : $\mathbb{Z}$
running_T1 : $\mathbb{B}$
running_T2 : $\mathbb{B}$
clock : pygame clock

17.4.2 State Invariant

N/A

17.4.3 Assumptions & Design Decisions

- This is to run the PVE mode for the game, the mode will call the other classes to create the map, the players' tanks, the enemy tanks, and the players' home. Players can win the game by staying alive(each player can reborn 2 times) for a certain time, and the players will lose if both of them die in that time or their home is destroyed.

17.4.4 Access Routine Semantics

PvsP():

- transition : The game_result, coordinate_T1, coordinate_T2, deadCount1 and deadCount2 are initialized to 1, (8,24), (16,24), 0 and 0, respectively, and the players' tanks are created when the by calling the choosenTankScreen() after the palyers selecting the tank. The Map file is read by calling the loadingMapScreen() and the map is created by call the Map module. The images and sound files are loaded into backgroup_image, home_image, home_destroyde_image, fire_sound, start_sound, respectively. The map is loaded by calling loadPVEMap(), and the food is created by calling Food(). allTankGroup and myTankGroup will initialized as the pygame sprite group and two players' tanks are added to the myTankGroup. Four events called MYBULLETCOOLINGEVENT, BULLETTP, LEAP, TIME are created and the timer are set to be 200ms, 1000ms, 1000ms and 1000ms, respectively. Delay, moving, movdir, moving2, movdir2, enemyNumber, enemyCouldMove, switch_R1_R2_image, homeDead, running_T1, running_T2 are initialized to 100, 0, 0, 0, 0, 3, True, True, 0, True, True, respectively, and the clock is created as a pygame clock.

- exception:
  None.

18 MIS of P vs P Module

18.1 Template Module

PvsP

18.2 Uses

Pygame
Bullet
BulletControl
MyTankControl
DoubleLifeTank
FastBulletTank
Food
HighSpeedTank
Map

18.3 Syntax

18.3.1 Exported Types

N/A

18.3.2 Exported Access Programs

Name	In	Out	Exceptions
PvsP	-	-	-

18.4 Semantics

18.4.1 State Variables

game_result : $\mathbb{Z}$
coordinate_T1 : CoordinateT
coordinate_T2 : CoordinateT
myTank_T1 : MyTank
myTank_T2 : MyTank
returnMap : Map
resolution ; $\mathbb{Z}$, $\mathbb{Z}$
screen : pygame display mode
background_image : pygame loading image
home_image : pygame loading image
home_destroyed_image : pygame loading image
bang_sound : pygame loading sound
fire_sound : pygame loading sound
start_sound : pygame loading sound
allTankGroup : pygame sprite group
myTankGroup : pygame sprite group
bgMap : PVPMap
prop : Food
deadCount1 : $\mathbb{Z}$
deadCount2 : $\mathbb{Z}$
strtime1 : String
strtime2 : String
deadline : datetime
now : datetime
subtime : $\mathbb{Z}$
ttime = decTime
MYBULLETNOTCOOLINGEVENT : pygame event
BULLETP : pygame event
LEAP : pygame event
Time : pygame event
delay : $\mathbb{Z}$
moving : $\mathbb{Z}$
movdir : $\mathbb{Z}$
moving2 : $\mathbb{Z}$
movdir2 : $\mathbb{Z}$
enemyNumber : $\mathbb{Z}$

enemyCouldMove : $\mathbb{B}$
switch_R1_R2_image : $\mathbb{B}$
homeDead : $\mathbb{Z}$
running_T1 : $\mathbb{B}$
running_T2 : $\mathbb{B}$
clock : pygame clock

18.4.2 State Invariant

N/A

18.4.3 Assumptions & Design Decisions

- This is to run the PVP mode for the game, the mode will call the other classes to create the map, the players' tanks, the enemy tanks, and the players' homes. Players can win the game by the destroy the other one's tank by 3 times or destroy his home, which is represented by an eagle. If a certain time is over and both players are alive, it will be a tie.

18.4.4 Access Routine Semantics

PvsP():

- transition : The game_result, coordinate_T1, coordinate_T2, deadCount1 and deadCount2 are initialized to 1, (8,24), (16,24), 0 and 0, respectively, and the players' tanks are created when the by calling the choosenTankScreen() after the palyers selecting the tank. The Map file is read by calling the loadingMapScreen() and the map is created by call the Map module. The images and sound files are loaded into backgroup_image, home_image, home_destroyde_image, fire_sound, start_sound, respectively. The map is loaded by calling loadPVPMap(), and the food is created by calling Food(). allTankGroup and myTankGroup will initialized as the pygame sprite group and two players' tanks are added to the myTankGroup. Four events called MYBULLETCOOLINGEVENT, BULLETTP, LEAP, TIME are created and the timer are set to be 200ms, 1000ms, 1000ms and 1000ms, respectively. Delay, moving, movdir, moving2, movdir2, enemyNumber, enemyCouldMove, switch_R1_R2_image, homeDead, running_T1, running_T2 are initialized to 100, 0, 0, 0, 0, 3, True, True, 0, True, True, respectively, and the clock is created as a pygame clock.

- exception:
  None.

19 MIS Map Editing Module

19.1 Template Module

MapEditing

19.2 Uses

pygame
MyTank
MyTankControl
Screen
display
positionType

19.3 Syntax

19.3.1 Exported Types

N/A

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
mapEditing	-	-	-
~~chooseTankScreenMap~~	-	-	-
~~buildWall~~	-	-	-

19.4 Semantics

19.4.1 State Variables

None

19.4.2 State Invariant

None

19.4.3 Assumptions & Design Decisions

- The mapEditing module will be used to create a PVP or PVE map and save it as a text file in the corresponding folder.

19.4.4 Access Routine Semantics

mapEditing():

- transition : The mapEditing function will generate a map editing mode for users to create their own map. In this module, a tank will be created and tank control will be used. The user can add brick using key "J" and add iron wall by pressing key "K", and remove walls by pressing "L". The coordinates will be represented by the tank

on the screen. If the mode ends, the created map will be saved as a text file in the PVPmap or PVEmap folder.

- exception:
  None.

~~chooseTankScreenMap(coordinate_T1):~~

- transition: ~~The chooseTankScreenMap function will generate a tank based on the given coordinates.~~
- exception:
  ~~None~~.

~~buildWall()~~:

- transition: ~~The buildWall function will get the user's key_pressed input and generate the corresponding actions which includes adding brick walls, adding iron walls, and delete walls.~~
- exception:
  ~~None~~.

20 MIS of Main Module

20.1 Template Module

main

20.2 Uses

N/A

20.3 Syntax

20.3.1 Exported Types

N/A

20.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

20.4 Semantics

20.4.1 State Variables

None

20.4.2 State Invariant

None

20.4.3 Assumptions & Design Decisions

- The main module will be the file which starts the whole game. Also main module will be the overall controller of the game. Users can get an access to PVE, PVP, or map editing mode through the main module.

20.4.4 Access Routine Semantics

main():

- transition : The main function will be used as the menu of the whole game, which includes three accesses: PVE mode, PVP mode, and map editing mode.
- exception:
  None.