SE 3XA3: MIS
Space Invaders

March 18, 2022

Team Information:

Team Number	Name	MACID
L03 G07	Qianlin Chen	chenq84
	Jiacheng Wu	wuj187
	Tingyu Shi	shit19

Table 1: Revision History

Date	**Developer(s)**	**Change**
January 26, 2022	All team members	Initial Document
March 18, 2022	Qianlin Chen	Display Modules
March 10, 2022	Jiacheng Wu	Control Modules
March 10, 2022	Tingyu Shi	Model modules

Contents

List of Tables

List of Figures

1 MonsterColor Module

Module

MonsterColor

Uses

None

Syntax

Exported Constants

None

Exported Types

MonsterType = $\{RED, BLUE, PINK\}$ # Represent Three colors of monsters

Exported Access Programs

None (This is an Enum class in python)

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

None

Consideration

When implementing in Python, use Enum class.

2 BulletState Module

Module

BulletState

Uses

None

Syntax

Exported Constants

None

Exported Types

BulletState = $\{FIRE, READY\}$ # Represent two states of bullets

Exported Access Programs

None (This is an Enum class in python)

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

None

Consideration

When implementing in Python, use Enum class.

3 Monster Module

Template Module

Monster

Uses

MonsterColor, Bullet

Syntax

Exported Constants

None

Exported Types

Monster = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Monster	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $MonterColor$	Monster	IllegalArgumentException
setX	$\mathbb{R}$		IllegalArgumentException
setY	$\mathbb{R}$		IllegalArgumentException
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
getColor		$MonsterColor$	
reduce_life			
isDead		$\mathbb{B}$	
move			
shoot		$Bullet$	

Semantics

State Variables

$speed$: $\mathbb{R}$
X: $\mathbb{R}$
Y: $\mathbb{R}$
$monster\_color$: $MonsterColor$
X_change: $\mathbb{R}$
Y_change: $\mathbb{R}$
$life$: $\mathbb{N}$

State Invariant

$0 \leq X \leq 736$

Assumptions

None

Access Routine Semantics

new Monster($s, x, y, color$):

- transition:
  $speed, X, Y, monster\_color, X\_change, Y\_change := s, x, y, color, s, 40$
  $color = MonsterColor.GREEN \Rightarrow life := 1$
  $color = MonsterColor.BLUE \Rightarrow life := 2$
  $color = MonsterColor.PINK \Rightarrow life := 3$
- output: $out := self$
- exception: exc := $((s < 0) \vee (x < 0) \vee (y < 0) \Rightarrow$ IllegalArgumentException)

setX(x):

- transition: $X := x$
- output: none
- exception: $((x < 0) \Rightarrow$ IllegalArgumentException)

setY(y):

- transition: $Y := y$
- output: none
- exception: $((y < 0) \Rightarrow$ IllegalArgumentException)

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

getColor():

- transition: none
- output: $out := monster\_color$

- exception: none

reduce_life():

- transition: $life := life - 1$
- output: none
- exception: none

isDead():

- transition: none
- output: $life = 0$
- exception: none

move():

- transition:
  $X := X + X\_change$
  $X \leq 0 \Rightarrow (X\_change,\ Y := speed,\ Y + Y\_change)$
  $X \geq 736 \Rightarrow (X\_change,\ Y := -1 * speed,\ Y + Y\_change)$
- output: none
- exception: none

shoot():

- transition: none
- output: $new\ Bullet(20,\ X,\ Y)$
- exception: none

4 SpaceShip Module

Template Module

SpaceShip

Uses

Bullet

Syntax

Exported Constants

None

Exported Types

SpaceShip = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Monster	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	SpaceShip	IllegalArgumentException
setX	$\mathbb{R}$		IllegalArgumentException
setY	$\mathbb{R}$		IllegalArgumentException
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
moveLeft			
moveRight			
stopMove			
reduce_life			
isDead		$\mathbb{B}$	
boundary_detection			
shoot		*Bullet*	
update			

Semantics

State Variables

speed: $\mathbb{R}$
X: $\mathbb{R}$
Y: $\mathbb{R}$
X_change: $\mathbb{R}$
life: $\mathbb{N}$

State Invariant

$0 \leq X \leq 736$
$0 \leq life \leq 5$

Assumptions

None

Access Routine Semantics

new SpaceShip(s, x, y):

- transition: $speed, X, Y, X\_change, life := s, x, y, 0, 5$
- output: $out := self$
- exception: exc := $((s < 0) \vee (x < 0) \vee (y < 0) \Rightarrow$ IllegalArgumentException)

setX(x):

- transition: $X := x$
- output: none
- exception: $((x < 0) \Rightarrow$ IllegalArgumentException)

setY(y):

- transition: $Y := y$
- output: none
- exception: $((y < 0) \Rightarrow$ IllegalArgumentException)

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

moveLeft():

- transition: $X\_change := -1 * speed$
- output: none

- exception: none

moveRight():

- transition: $X\_change := speed$
- output: none
- exception: none

stopMove():

- transition: $X\_change := 0$
- output: none
- exception: none

reduce_life():

- transition: $life := life - 1$
- output: none
- exception: none

isDead():

- transition: none
- output: $life = 0$
- exception: none

boundary_detection():

- transition:
  $X \leq 0 \Rightarrow (X\_change := 0)$
  $X \geq 736 \Rightarrow (X\_change := 736)$
- output: none
- exception: none

shoot():

- transition: none
- output: $new\ Bullet(20,\ X,\ Y)$
- exception: none

update():

- transition: $X := X + X\_change$
- output: none
- exception: none

5 Bullet Module

Template Module

Bullet

Uses

none

Syntax

Exported Constants

None

Exported Types

Bullet = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Bullet	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	Bullet	IllegalArgumentException
setX	$\mathbb{R}$		IllegalArgumentException
setY	$\mathbb{R}$		IllegalArgumentException
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
setState	*BulletState*		
getState		*BulletState*	
move			

Semantics

State Variables

speed: $\mathbb{R}$
X: $\mathbb{R}$
Y: $\mathbb{R}$
Y_change: $\mathbb{R}$
state: *BulletState*

State Invariant

none

Assumptions

None

Access Routine Semantics

new Bullet(s, x, y):

- transition: $speed, X, Y, Y\_change, state := s, x, y, s, BulletState.READY$
- output: $out := self$
- exception: exc := $((s < 0) \vee (x < 0) \vee (y < 0) \Rightarrow$ IllegalArgumentException)

setX(x):

- transition: $X := x$
- output: none
- exception: $((x < 0) \Rightarrow$ IllegalArgumentException)

setY(y):

- transition: $Y := y$
- output: none
- exception: $((y < 0) \Rightarrow$ IllegalArgumentException)

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

setState($newState$):

- transition: $state := newState$
- output: none
- exception: none

getState():

- transition: none
- output: $out := state$
- exception: none

move():

- transition:
  $Y \leq 0 \Rightarrow Y,\ state := 530,\ BulletState.READY$
  $state = BulletState.FIRE \Rightarrow Y := Y - Y\_change$
- output: none
- exception: none

6 Score Module

Template Module

Score

Uses

none

Syntax

Exported Constants

None

Exported Types

Score = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new score		Score	
getScore		$\mathbb{N}$	
increaseAmount	$\mathbb{N}$		

Semantics

State Variables

score: $\mathbb{N}$

State Invariant

$score \geq 0$

Assumptions

None

Access Routine Semantics

new Score():

- transition: $score := 0$
- output: $out := self$
- exception: exc := none

getScore():

- transition: none
- output: $out := score$
- exception: none

increaseAmount($amount$):

- transition: $score := score + amount$
- output: none
- exception: none

7 Obstacle Module

Template Module

Obstacle

Uses

none

Syntax

Exported Constants

None

Exported Types

Obstacle = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Obstacle	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	Obstacle	
reduce_area			
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
getArea		$\mathbb{R}$	

Semantics

State Variables

X: $\mathbb{R}$
Y: $\mathbb{R}$
Width: $\mathbb{R}$
Height: $\mathbb{R}$
Area: $\mathbb{R}$

State Invariant

all state variables ≥ 0

Assumptions

None

Access Routine Semantics

new Obstacle(x, y, w, h):

- transition: $X, Y, Width, Height, Area := x,\ y,\ w,\ h,\ w * h$
- output: $out := self$
- exception: exc := none

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

getArea():

- transition: none
- output: $out := Area$
- exception: none

reduce_area($amount$):

- transition: $Area := Area - amount$
- output: none
- exception: none

8 Ammo Module

Template Module

Ammo

Uses

None

Syntax

Exported Constants

None

Exported Types

Ammo = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Ammo	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	Ammo	IllegalArgumentException
setX	$\mathbb{R}$		IllegalArgumentException
setY	$\mathbb{R}$		IllegalArgumentException
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
reduce_life			
isDead		$\mathbb{B}$	
move			

Semantics

State Variables

$speed$: $\mathbb{R}$
X: $\mathbb{R}$
Y: $\mathbb{R}$
X_change: $\mathbb{R}$
Y_change: $\mathbb{R}$
$life$: $\mathbb{N}$

State Invariant

$0 \leq X \leq 736$

Assumptions

None

Access Routine Semantics

new Bomb(s, x, y):

- transition:
  $speed, X, Y, X\_change, Y\_change, life := s, x, y, s, 40, 1$
- output: $out := self$
- exception: exc := $((s < 0) \vee (x < 0) \vee (y < 0) \Rightarrow$ IllegalArgumentException)

setX(x):

- transition: $X := x$
- output: none
- exception: $((x < 0) \Rightarrow$ IllegalArgumentException)

setY(y):

- transition: $Y := y$
- output: none
- exception: $((y < 0) \Rightarrow$ IllegalArgumentException)

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

reduce_life():

- transition: $life := life - 1$
- output: none
- exception: none

isDead():

- transition: none
- output: $life = 0$

- exception: none

move():

- transition:
  $X := X + X\_change$
  $X \leq 0 \Rightarrow (X\_change,\ Y := speed,\ Y + Y\_change)$
  $X \geq 736 \Rightarrow (X\_change,\ Y := -1 * speed,\ Y + Y\_change)$
- output: none
- exception: none

9 Bomb Module

Template Module

Bomb

Uses

None

Syntax

Exported Constants

None

Exported Types

Bomb = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Bomb	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	Bomb	IllegalArgumentException
setX	$\mathbb{R}$		IllegalArgumentException
setY	$\mathbb{R}$		IllegalArgumentException
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
reduce_life			
isDead		$\mathbb{B}$	
move			

Semantics

State Variables

$speed$: $\mathbb{R}$
X: $\mathbb{R}$
Y: $\mathbb{R}$
X_change: $\mathbb{R}$
Y_change: $\mathbb{R}$
$life$: $\mathbb{N}$

State Invariant

$0 \leq X \leq 736$

Assumptions

None

Access Routine Semantics

new Ammo(s, x, y):

- transition:
  $speed, X, Y, X\_change, Y\_change, life := s, x, y, s, 40, 1$
- output: $out := self$
- exception: exc := $((s < 0) \vee (x < 0) \vee (y < 0) \Rightarrow$ IllegalArgumentException)

setX(x):

- transition: $X := x$
- output: none
- exception: $((x < 0) \Rightarrow$ IllegalArgumentException)

setY(y):

- transition: $Y := y$
- output: none
- exception: $((y < 0) \Rightarrow$ IllegalArgumentException)

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

reduce_life():

- transition: $life := life - 1$
- output: none
- exception: none

isDead():

- transition: none
- output: $life = 0$

- exception: none

move():

- transition:
  $X := X + X\_change$
  $X \leq 0 \Rightarrow (X\_change,\ Y := speed,\ Y + Y\_change)$
  $X \geq 736 \Rightarrow (X\_change,\ Y := -1 * speed,\ Y + Y\_change)$
- output: none
- exception: none

10 Heart Module

Template Module

Heart

Uses

None

Syntax

Exported Constants

None

Exported Types

Heart = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Heart	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	Heart	IllegalArgumentException
setX	$\mathbb{R}$		IllegalArgumentException
setY	$\mathbb{R}$		IllegalArgumentException
getX		$\mathbb{R}$	
getY		$\mathbb{R}$	
reduce_life			
isDead		$\mathbb{B}$	
move			

Semantics

State Variables

$speed$: $\mathbb{R}$
X: $\mathbb{R}$
Y: $\mathbb{R}$
X_change: $\mathbb{R}$
Y_change: $\mathbb{R}$
$life$: $\mathbb{N}$

State Invariant

$0 \leq X \leq 736$

Assumptions

None

Access Routine Semantics

new Heart(s, x, y):

- transition:
  $speed, X, Y, X\_change, Y\_change, life := s, x, y, s, 40, 1$
- output: $out := self$
- exception: exc := $((s < 0) \vee (x < 0) \vee (y < 0) \Rightarrow$ IllegalArgumentException)

setX(x):

- transition: $X := x$
- output: none
- exception: $((x < 0) \Rightarrow$ IllegalArgumentException)

setY(y):

- transition: $Y := y$
- output: none
- exception: $((y < 0) \Rightarrow$ IllegalArgumentException)

getX():

- transition: none
- output: $out := X$
- exception: none

getY():

- transition: none
- output: $out := Y$
- exception: none

reduce_life():

- transition: $life := life - 1$
- output: none
- exception: none

isDead():

- transition: none
- output: $life = 0$

- exception: none

move():

- transition:
  $X := X + X\_change$
  $X \leq 0 \Rightarrow (X\_change,\ Y := speed,\ Y + Y\_change)$
  $X \geq 736 \Rightarrow (X\_change,\ Y := -1 * speed,\ Y + Y\_change)$
- output: none
- exception: none

11 CollisionDectection Module

Service Module

Service

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
isCollided	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$	$\mathbb{B}$	

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

isCollided (x_1, x_2, y_1, y_2):

- transition: none
- output: $distance(x_1, x_2, y_1, y_2) \leq 27$
- exception: exc := none

Local Function

distance: $[\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}] \Rightarrow \mathbb{R}$
distance$(x_1, x_2, y_1, y_2) \equiv \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

12 MonsterMatrix Module

Template Module

MonsterMatrix

Uses

Monster, Ammo, Heart, Bomb

Syntax

Exported Constants

None

Exported Types

MonsterMatrix = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new MonsterMatrix	$\mathbb{N}$, $\mathbb{R}$	MonsterMatrix	IllegalArgumentException
getMatrix		seq of seq [Monster, Ammo, Heart, Bomb]	
move			
shoot			

Semantics

State Variables

$speed$: $\mathbb{R}$
M: $seq\ of\ seq[Ammo\ Monster\ Heart\ Bomb]$

State Invariant

None

Assumptions

None

Access Routine Semantics

new MonsterMatrix($round, s$):

- transition:
  $speed := s$
  $round = 1 \Rightarrow M := m1(with\ monsters\ randomly\ replaced\ by\ Ammo,\ Bomb,\ Heart)$
  $round = 2 \Rightarrow M := m2(with\ monsters\ randomly\ replaced\ by\ Ammo,\ Bomb,\ Heart)$
  $round = 3 \Rightarrow M := m3(with\ monsters\ randomly\ replaced\ by\ Ammo,\ Bomb,\ Heart)$
  $round = 4 \Rightarrow M := m4(with\ monsters\ randomly\ replaced\ by\ Ammo,\ Bomb,\ Heart)$
  $round = 5 \Rightarrow M := m5(with\ monsters\ randomly\ replaced\ by\ Ammo,\ Bomb,\ Heart)$
- output: $out := self$
- exception: exc := $(\neg(0 \leq round \leq 5) \vee (s < 0)) \Rightarrow$ IllegalArgumentException)

getMatrix():

- transition: none
- output: $out := M$
- exception: none

move():

- transition: Monster Matrix moves in direction east → south → west
- output: none
- exception: none

shoot():

- transition: Monsters in the first row of M shoot bullets randomly.
- output: none
- exception: none

Local Types

$$\text{m1} \equiv \begin{bmatrix} GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \end{bmatrix}$$

$$\text{m2} \equiv \begin{bmatrix} BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \end{bmatrix}$$

$$\text{m3} \equiv \begin{bmatrix} BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \end{bmatrix}$$

$$\text{m4} \equiv \begin{bmatrix} PM & PM & PM & PM & PM & PM & PM & PM & PM & PM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ BM & BM & BM & BM & BM & BM & BM & BM & BM & BM \\ GM & GM & GM & GM & GM & GM & GM & GM & GM & GM \end{bmatrix}$$

$$\text{m5} \equiv \begin{bmatrix} PM & PM & PM & PM & PM & PM & PM & PM & PM & PM \\ PM & PM & PM & PM & PM & PM & PM & PM & PM & PM \\ PM & PM & PM & PM & PM & PM & PM & PM & PM & PM \\ PM & PM & PM & PM & PM & PM & PM & PM & PM & PM \\ PM & PM & PM & PM & PM & PM & PM & PM & PM & PM \end{bmatrix}$$

GM means green monster.
BM means blue monster.
PM means pink monster.

13 MonsterDisplay Module

UserInterface Module

MonsterDisplay

Uses

Monster, MonsterColor

Syntax

Exported Constants

N/A

Exported Types

MonsterDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new MonsterDisplay	pygame window object, MonsterColor	MonsterDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, Boolean		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: *pygame window object*
img : Monster Picture

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new MonsterDisplay($screen, monster\_color$)

- transition:
  $SCREEN := screen$
  $monster\_color = MonsterColor.GREEN \Rightarrow img :=$ green monster picture
  $monster\_color = MonsterColor.BLUE \Rightarrow img :=$ blue monster picture
  $monster\_color = MonsterColor.PINK \Rightarrow img :=$ pink monster picture
- output: $out := self$
- exception: None

show($x, y, isDead$):

- transition: $isDead \Rightarrow Display\ img\ at\ (x, y)$
- output: none
- input definitions: x and y represent the monster display coordinate. $isDead$ is used to clarify whether the monster is killed.
- exception: None

14 SpaceShipDisplay Module

UserInterface Module

SpaceShipDisplay

Uses

SpaceShip

Syntax

Exported Constants

N/A

Exported Types

SpaceShipDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new SpaceShipDisplay	pygame window object, $\mathbb{N}$	SpaceShipDisplay	
show	$\mathbb{N}$, $\mathbb{N}$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

N/A

State Invariant

$SCREEN$: *pygame window object*
img : SpaceShip Picture

Assumptions

N/A

Access Routine Semantics

new SpaceShipDisplay($screen, spaceship\_number$)

- transition:
  $SCREEN := screen$
  $space\_number = 1 \Rightarrow img :=$ spaceship1 picture
  $space\_number = 2 \Rightarrow img :=$ spaceship2 picture
- output: $out := self$
- exception: None

show(x, y):

- transition: Display img at (x, y)
- input definitions: x and y represent the spaceship display coordinate.
- output: $out := self$
- exception: None

15 BulletDisplay Module

UserInterface Module

BulletDisplay

Uses

BulletState, Bullet

Syntax

Exported Constants

N/A

Exported Types

BulletDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new BulletDisplay	pygame window object	BulletDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, *BulletState*		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: *pygame window object*
img : Bullet Picture

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new BulletDisplay($screen$)

- transition:
  $SCREEN := screen$
  $img :=$ bullet picture
- output: $out := self$
- exception: None

show($x, y, state$):

- transition: $state = BulletState.FIRE \Rightarrow$ Display img *at* (x, y)
- input definitions: x and y represent the bullet's original coordinate and it will increase if the bullet state is FIRE.
- output: none
- exception: None

16 ScoreDisplay Module

UserInterface Module

ScoreDisplay

Uses

Score

Syntax

Exported Constants

N/A

Exported Types

ScoreDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ScoreDisplay	pygame window object	ScoreDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{N}$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: *pygame window object*

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new ScoreDisplay($screen$)

- transition: $SCREEN := screen$
- output: $out := self$
- exception: None

show($x, y, score$):

- transition: Display $score$ at (x, y).
- input definitions: x and y represent the score's coordinate. $score$ represent the total scores of player(s).
- output: none
- exception: None

17 ObstaclesDisplay Module

UserInterface Module

ObstaclesDisplay

Uses

Obstacle

Syntax

Exported Constants

N/A

Exported Types

ObstaclesDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ObstaclesDisplay	pygame window object	ObstaclesDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{B}$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: *pygame window object*
img : Obstacle Picture

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new ObstaclesDisplay($screen$)

- transition:
  $SCREEN := screen$
  $img :=$ Obstacle Picture
- output: $out := self$
- exception: None

show($x, y, isDestroy$):

- transition: $\neg isDestroy \Rightarrow Display\ img\ at\ (x, y)$
- input definitions: x and y represent the obstacles' coordinate. $isDestroy$ represent the state of obstacles, 'True' if the obstacles are destroy otherwise 'False'.
- output: none
- exception: None

18 AmmoDisplay Module

UserInterface Module

AmmoDisplay

Uses

Ammo

Syntax

Exported Constants

N/A

Exported Types

AmmoDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new AmmoDisplay	pygame window object	AmmoDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{B}$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: pygame window object
img : Picture of Ammo

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new AmmoDisplay(*screen*)

- transition:
  $SCRREN := screen$
  $img := Ammo\ Picture$
- output: $out := self$
- exception: None

show(x, y, isShot):

- transition: $\neg isShot \Rightarrow Display\ img\ at\ (x, y)$
- input definitions: x, y represent the coordinate of ammo picture to be displayed. *isShot* represents the state of ammo, it is True if the ammo is shoot.
- output: none
- exception: None

19 HeartDisplay Module

UserInterface Module

HeartDisplay

Uses

Heart

Syntax

Exported Constants

N/A

Exported Types

HeartDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new HeartDisplay	pygame window object	HeartDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{B}$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: pygame window object
img : Picture of Heart

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new HeartDisplay(*screen*)

- transition:
  $SCRREN := screen$
  $img := Heart\ Picture$
- output: $out := self$
- exception: None

show(x, y, isShot):

- transition: $\neg isShot \Rightarrow Display\ img\ at\ (x, y)$
- input definitions: x, y represent the coordinate of Heart picture to be displayed. *isShot* represents the state of Heart, it is True if the Heart is shoot.
- output: none
- exception: None

20 BombDisplay Module

UserInterface Module

BombDisplay

Uses

Bomb

Syntax

Exported Constants

N/A

Exported Types

BombDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new BombDisplay	pygame window object	BombDisplay	
show	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{B}$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: pygame window object
img : Picture of Bomb

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new BombDisplay($screen$)

- transition:
  $SCRREN := screen$
  $img := Bomb\ Picture$
- output: $out := self$
- exception: None

show(x, y, isShot):

- transition: $\neg isShot \Rightarrow Display\ img\ at\ (x, y)$
- input definitions: x, y represent the coordinate of Bomb picture to be displayed. $isShot$ represents the state of Bomb, it is True if the Bomb is shoot.
- output: none
- exception: None

21 MonsterMatrixDisplay Module

UserInterface Module

MonsterMatrixDisplay

Uses

BombDisplay, MonsterDisplay, AmmoDisplay, HeartDisplay, MonsterMatrix

Syntax

Exported Constants

N/A

Exported Types

MonsterMatrixDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new MonsterMatrixDisplay	pygame window object	MonsterMatrixDisplay	
show	$MonsterMatrix$		

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: pygame window object

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new MonsterMatrixDisplay($screen$)

- transition: $SCREEN := screen$
- output: $out := self$
- exception: None

show(M):

- transition: $\forall object \in M \mid object.show(x,\ y,\ isDead/isShot)$
  M here could be $Monster\ Ammo\ Heart\ Bomb$
- output: none
- exception: None

22 SetUpDisplay Module

UserInterface Module

SetUpDisplay

Uses

None

Syntax

Exported Constants

N/A

Exported Types

SetUpDisplay = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new SetUpDisplay	screen	SetUpDisplay	
show			
getScreen		pygame window object	

Semantics

Environment Variables

screen: It is the game screen that is manipulated by the following functions to alter display. This is update by a precise frame rate to depict various game objects on the game screen.

State Variables

$SCREEN$: pygame window object

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

new SetUpDisplay()

- transition: $SCREEN :=$ *new pygame window object*
- output: $out := self$
- exception: None

show():

- transition: Display The following contents
  - Welcoming message
  - Display game mode selection
  - Game instruction
  - Prevent game addiction message
- output: none
- exception: None

getScreen()

- transition: none
- output: $out := SCREEN$
- exception: none

23 SingleController Module

Template Module

SingleController

Uses

BulletDisplay, MonsterMatrixDisplay, SpaceShipDisplay, ScoreDisplay, ObstaclesDisplay

Syntax

Exported Constants

None

Exported Types

SingleController = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new SingleController		SingleController	
run	Keyboard Inputs		

Semantics

State Variables

All the model objects and corresponding display objects.

State Invariant

None

Assumptions

None

Access Routine Semantics

new SingleController():

- transition: Create the model objects and corresponding display objects.
- output: *out* := *self*
- exception: None

run()

- transition: The controller should do the following things:
  - Let player move space by pressing $\leftarrow$ and $\rightarrow$.
  - Let player shoot bullet by pressing `SPACE`.
  - If any monster is dead or any game item is shot, let them disappear from the game window.
  - If any monsters are shot, increase the score.
  - If any game items are shot, do corresponding operations.
  - If a round is finished, go to the next round.
  - If the spaceship is shot, decrease spaceship lives.
  - If the obstacle is shot, decrease obstacle areas.
- output: none
- exception: none

24 DoubleController Module

Template Module

DoubleController

Uses

BulletDisplay, MonsterMatrixDisplay, SpaceShipDisplay, ScoreDisplay, ObstaclesDisplay

Syntax

Exported Constants

None

Exported Types

DoubleController = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new DoubleController		DoubleController	
run	Keyboard Inputs		

Semantics

State Variables

All the model objects and corresponding display objects.

State Invariant

None

Assumptions

None

Access Routine Semantics

new DoubleController():

- transition: Create the model objects and corresponding display objects.
- output: *out* := *self*
- exception: None

run()

- transition: The controller should do the following things:
  - Let player1 move space by pressing ← and →.
  - Let player1 shoot bullet by pressing `SPACE`.
  - Let player2 move space by pressing `A` and `D`.
  - Let player2 shoot bullet by pressing `S`.
  - If any monster is dead or any game item is shot, let them disappear from the game window.
  - If any monsters are shot, increase the score.
  - If any game items are shot, do corresponding operations.
  - If a round is finished, go to the next round.
  - If the spaceship is shot, decrease spaceship lives.
  - If the obstacle is shot, decrease obstacle areas.
- output: none
- exception: none

25 TotalController Module

Template Module

TotalController

Uses

SetUpDisplay

Syntax

Exported Constants

None

Exported Types

TotalController = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new TotalController		TotalController	
run	Keyboard Input		

Semantics

State Variables

s : SetUpDisplay

State Invariant

None

Assumptions

None

Access Routine Semantics

new DoubleController():

- transition: $s := newSetUpDisplay()$
- output: $out := self$
- exception: None

run()

- transition:
  $s.run()$
  If user chooses single player model ⇒ Invoke SingleController
  If user chooses double player model ⇒ Invoke DoubleController

- output: none
- exception: none

26 Driver Module

Template Module

Driver

Uses

TotalController

Syntax

Exported Constants

None

Exported Types

Driver = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Driver		Driver	
run			

Semantics

State Variables

t : TotalController

State Invariant

None

Assumptions

None

Access Routine Semantics

new Driver():

- transition: $t := new\ TotalController()$
- output: $out := self$
- exception: None

run()

- transition: $t.run()$
- output: none
- exception: none