SE 3XA3: Module Interface Specification
Snake Game Remake

Team #, 302, Team Name: 404
Student: Shunbo Cui cuis13
Student: Xiangxin Kong kongx9
Student: Shuo Zhang zhans18

March 13, 2020

Table 1: Revision History

Date	**Members**	**Change**
March 12, 2020	Shuo Zhang, Xiangxin Kong, Shunbo Cui	Create the MIS
...	...	...

1 Map Module

Module

Map

Uses

None

Syntax

Exported Constants

map1 : {(227, 351, 16, 16), (243, 351, 16, 16), (259, 351, 16, 16), (275, 351, 16, 16), (291, 351, 16, 16), (307, 351, 16, 16), (323, 351, 16, 16), (339, 351, 16, 16), (355, 351, 16, 16), (371, 351, 16, 16), (387, 351, 16, 16), (403, 351, 16, 16), (419, 351, 16, 16), (435, 351, 16, 16), (451, 351, 16, 16), (819, 351, 16, 16), (803, 351, 16, 16), (787, 351, 16, 16), (771, 351, 16, 16), (755, 351, 16, 16), (739, 351, 16, 16), (723, 351, 16, 16), (707, 351, 16, 16), (691, 351, 16, 16), (675, 351, 16, 16), (659, 351, 16, 16), (643, 351, 16, 16), (627, 351, 16, 16), (611, 351, 16, 16), (595, 351, 16, 16), (515, 127, 16, 16), (515, 143, 16, 16), (515, 159, 16, 16), (515, 175, 16, 16), (515, 191, 16, 16), (515, 207, 16, 16), (515, 223, 16, 16), (515, 239, 16, 16), (515, 255, 16, 16), (515, 271, 16, 16), (515, 575, 16, 16), (515, 559, 16, 16), (515, 543, 16, 16), (515, 527, 16, 16), (515, 511, 16, 16), (515, 495, 16, 16), (515, 479, 16, 16), (515, 463, 16, 16), (515, 447, 16, 16), (515, 431, 16, 16)}

map2 : (227, 351, 16, 16), (243, 351, 16, 16), (259, 351, 16, 16), (275, 351, 16, 16), (291, 351, 16, 16), (819, 351, 16, 16), (803, 351, 16, 16), (787, 351, 16, 16), (771, 351, 16, 16), (755, 351, 16, 16), (483, 127, 16, 16), (483, 143, 16, 16), (483, 159, 16, 16), (483, 175, 16, 16), (483, 191, 16, 16), (483, 207, 16, 16), (483, 223, 16, 16), (483, 239, 16, 16), (483, 255, 16, 16), (483, 271, 16, 16), (483, 287, 16, 16), (483, 303, 16, 16), (483, 319, 16, 16), (483, 335, 16, 16), (483, 351, 16, 16), (483, 367, 16, 16), (483, 383, 16, 16), (483, 399, 16, 16), (483, 415, 16, 16), (483, 431, 16, 16), (483, 447, 16, 16), (483, 463, 16, 16), (483, 479, 16, 16), (547, 575, 16, 16), (547, 559, 16, 16), (547, 543, 16, 16), (547, 527, 16, 16), (547, 511, 16, 16), (547, 495, 16, 16), (547, 479, 16, 16), (547, 463, 16, 16), (547, 447, 16, 16), (547, 431, 16, 16), (547, 415, 16, 16), (547, 399, 16, 16), (547, 383, 16, 16), (547, 367, 16, 16), (547, 351, 16, 16), (547, 335, 16, 16), (547, 319, 16, 16), (547, 303, 16, 16), (547, 287, 16, 16), (547, 271, 16, 16), (547, 255, 16, 16), (547, 239, 16, 16), (547, 223, 16, 16)

Exported Types

None

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None

2 Snake Module

Snake

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new snake		snake	
setDirectione	$\mathbb{Z}$		
eat			
move			
getSnakeBody		set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$	
getLength		$\mathbb{Z}$	
getDirection		$\mathbb{Z}$	
getHead		$(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$	

Semantics

State Variables

snakebody: set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$
direction: $\mathbb{Z}$

State Invariant

DEFAULT_SNAKE_LENGTH = 5
DEFAULT_SNAKE_DIRECTION = 3

Assumptions

- The constructor snake is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.
- direction can only be 0, 1, 2 or 3.

Access Routine Semantics

snake():

- transition: direction := DEFAULT_SNAKE_DIRECTION, snakebody := ($\forall i : 0 \leq i \leq$ DEFAULT_SNAKE_LENGTH | (355 - i * 16, 191, 16, 16) $\in$ snakebody)
- output: *out* := *self*
- exception: None

setDirection(dir):

- transition: ((direction $\geq 3 \wedge$ dir < 3) $\vee$ (direction $\leq 2 \wedge$ dir > 2)) $\implies$ direction := dir
- ouput := None
- exception: None

eat():

- transition: snakebody := snakebody + last $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$ element of snakebody before the move [$\leftarrow$ *this element is cut off from the snakebody after every move() operation*]
- ouput := None

- exception: None

getSnakeBody():

- transition: None
- ouput := snakebody
- exception: None

getLength():

- transition: None
- ouput := size of snakebody
- exception: None

getDirection():

- transition: None
- ouput := direction
- exception: None

getHead():

- transition: None
- ouput := first $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$ element of snakebody
- exception: None

move():

- transition: $(\forall$ i $\in \mathbb{Z}|$ size of snakebody $\geq$ i ≥ 1 : the ith element of snakebody := the (i-1)th element of snakebody) , ((direction = 1 $\implies$ decreaseY()) $\vee$ (direction = 2 $\implies$ increaseY()) $\vee$ (direction = 3 $\implies$ increaseX()) $\vee$ (direction = 4 $\implies$ decreaseX()))
- ouput := None
- exception: None

Local Functions

increaseY: set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}) \rightarrow$ set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$
increaseY() $\equiv$ 1st element snakebody := $(\mathbb{R}, \mathbb{R} + 16, \mathbb{R}, \mathbb{R})$ of 2nd element of snakebody

decreaseY: set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}) \rightarrow$ set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$
decreaseY() $\equiv$ 1st element snakebody := $(\mathbb{R}, \mathbb{R} - 16, \mathbb{R}, \mathbb{R})$ of 2nd element of snakebody

decreaseX: set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}) \rightarrow$ set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$
decreaseX() $\equiv$ 1st element snakebody := $(\mathbb{R} - 16, \mathbb{R}, \mathbb{R}, \mathbb{R})$ of 2nd element of snakebody

increaseX: set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}) \rightarrow$ set of $(\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R})$
increaseX() $\equiv$ 1st element snakebody := $(\mathbb{R} + 16, \mathbb{R}, \mathbb{R}, \mathbb{R})$ of 2nd element of snakebody

3 Main Module

Main Screen

Uses

Game board

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Main			
new MainScreen		MainScreen	

Semantics

State Variables

gameboard $\in$ GameBoardPanel

Environment Variables

keyboard button: up, down, left ,right, space, esc
mouse: leftclick

State Invariant

levelStrings = { "Easy", "Normal", "Hard" }

Assumptions

Main() will run before any other programs.

Access Routine Semantics

Main():

- transition: call Mainscreen()
- output: None
- exception: None

Mainscreen():

- transition: generate a visible window with level selection buttons, a special mode selection button and snake color selection buttons. (select "Easy" $\implies$ call GameBoardPanel(1)) $\vee$ (select "Normal" $\implies$ call GameBoardPanel(2)) $\vee$ (select "Hard" $\implies$ call GameBoardPanel(3))
- ouput := *out* := *self*
- exception: None

4 Sound Module

SoundManger

Uses

None

Syntax

Exported Constants

None

Exported Types

Files: bgm_file = start.wav; eatfood_file = food.wav; eatitem_file = item.wav; collide_file = collision.wav.

Exported Access Programs

Routine name	In	Out	Exceptions
new SoundManger	String	SoundManger	
startSound			
pauseSound			
stopSound			

Semantics

State Variables

None

Environment Variables

State Invariant

None

Assumptions

SoundManger will run before any other programs in this module.

Access Routine Semantics

SoundManger(soundFilePath):

- transition: Open the sound effect that is stored at path *soundFilePath* and ready to play.
- ouput := *out* := *self*
- exception: None

startSound():

- transition: play the sound effect.
- ouput : None
- exception: None

pauseSound():

- transition: suspend the sound effect.
- ouput : None
- exception: None

stopSound():

- transition: terminate the sound effect.
- ouput : None
- exception: None

5 MIS of Food and Item Module

5.1 Interface Syntax

5.1.1 Exported Access Programs

Name	In	Out	Exceptions
generateFood	-	-	-
getFood	-	(R, R)	-
generateItem	-	-	-
getItem	-	(R, R)	-

5.2 Interface Semantics

5.2.1 State Variables

food: (R, R) - represents the current location of the food
item: (R, R) - represents the current location of the item

5.2.2 Assumptions

Other modules always call generateFood() before calling getFood()
Other modules always call generateItem() before calling getItem()

5.2.3 Access Program Semantics

generateFood():
transition: item:=pair of random integers from 0 to 227

generateItem():
transition: food:=pair of random integers from 0 to 227

getFood():
Output:= food

getItem():
Output:=item

6 GameBoard Module

Uses

Snake, Food and Item, Sound, Map

Syntax

Exported Constants

6.1 Interface Syntax

6.1.1 Exported Access Programs

Name	In	Out	Exceptions
keyPressed	event	-	-
InputManger	event	-	-
checkCollision	-	$\mathbb{B}$	-
drawMap	-	-	-
drawSnake	snake	-	-
DrawSnakeFood	food	-	-
DrawStatusbar	-	-	-

6.2 Interface Semantics

6.2.1 State Variables

food: $(\mathbb{R}, \mathbb{R})$ - represents the current location of the food
item: $(\mathbb{R}, \mathbb{R})$ - represents the current location of the item

isGameOver:$\mathbb{B}$ - repersents is the game over playerScore:$\mathbb{R}$ -repersents the current score snake: SNAKE- repersents the current snake

6.2.2 Assumptions

Variables should be set before trying to access them.
If no event is chosen, checkEvent returns a default value 0
If currState is 0, drawInterface does not change

6.2.3 Access Program Semantics

InputManger(gb):

- transition: gameboard := gb
- output: *out* := *self*
- exception: None

keyPressed(e):

- transition: (e = up button $\implies$ gameBoard.changeSnakeDirection(1)) $\vee$ (e = down button $\implies$ gameBoard.changeSnakeDirection(2)) $\vee$ (e = right button $\implies$ gameBoard.changeSnakeDirection(3)) $\vee$ (e = left button $\implies$ gameBoard.changeSnakeDirection(4)) $\vee$ (e = space $\wedge$ gameBoard.isGameRunning()=true $\implies$ gameBoard.pauseGame()) $\vee$ (e = space $\wedge$ gameBoard.isGameRunning()=false $\implies$ gameBoard.startGame()) $\vee$ (e = esc $\implies$ exit the game.)
- ouput := None
- exception: None

DrawSnake(a):

- Input: snake object
- draws the snake corresponding to the current state to the output window
- exception: None

DrawMap(a):

- Input: None
- draws the map corresponding to the map constant to the output windows

- exception: None

DrawSnakeFood(a):

- Input: food
- draws the snake food corresponding to the current food state to the map
- exception: None

DrawStatusbar():

- Input: None
- Draws the instructions, Show playscore, (Game over $\implies$ "Game over message at center")
- exception: None