Assignment 4 Specification and Requirements Documents

Omar Alkersh - alkersho

April 1, 2020

The specification for *Two Dots* consists of `Controller`, the main controller in the CV, `View`, the module responsible for the view, `BoardT`, the module holding the board information, `GameState` interface, the interface responsible for the win and lose conditions, and the three implementing modules of `GameState`.

The game is a 2 dimensional board which consists of coloured dots. The main action of the game is to connect at least two adjacent dots, not diagonally. Connecting dots consumes them, lowering the above dots to fill the gap then populates the empty top spots with new random dots. The win and lose conditions are defined by the game type.

There are three specified game types:

- Timer:
  - Gives the player a timer until the game is finished. The player aims to achieve the highest score during the time by connecting dot.
- Score:
  - The player tries to achieve a set score. When achieved the game ends and the player "wins"
- Moves:
  - The player has a limited number of moves before the game is over. During which he have to score a set score else he loses.

1 DONE Board Module

Template Module

BoardT

Uses

GameState

Syntax

Exported Constants

SIZE = 6

Exported Types

Colour = {Blue, Green, Pink, Red, Orange}
BoardT = ?

Exported Access Program

Routine name	In	Out	Exception
BoardT	GameState	BoardT	
consume	set(tuple(i: $\mathbb{N}$, j: $\mathbb{N}$))		IndexOutOfBoundsException, IllegalArgumentException
updateBoard			
getCell	$\mathbb{N}$, $\mathbb{N}$	Colour	IndexOutOfBoundsException

Semantics

- State Variables

  b: seq (seq (Colour))

  state: GameState

- State Invariant

  $|b| = SIZE$

  $\forall i, j : i, j \in \mathbb{N} \wedge 0 \leq i, j < |b| \bullet |b[i]| = |b[j]| \wedge |b[i]| = SIZE$

- Assumptions

  BoardT is called before any other routine is called.

- Access Routine Semantics

  `new` BoardT(gs):

  - transition: $state, b := gs, board$ such that $\forall i, j : i, j \in \mathbb{N} \wedge 0 \leq i, j < SIZE \bullet board[i][j] \neq Null$
  - output: $out := self$
  - exception: None

  consume(set):

- transition: The set of "coordinates" are set to `Null`, then it calls state.update(|set|, getCell(c)) where $c \in set$.
  - Coordinates are entries in the 2D array.
  - Down means that the object moves to the row below. If an object is at s[i][j], where $i \neq 0$ then moving down will put it at s[i-1][j].
- output: None
- Exception: $exc := \exists c : c \in set \bullet (0 \leq c.i < SIZE \land 0 \leq c.j < SIZE) \implies IndexOutOfBoundsException \mid \neg\forall i, j, k, l : i, j, k, l \in \mathbb{N} \land i < SIZE \land j < SIZE \land k < SIZE \land l < SIZE \bullet b[i][j] = b[k][l] \implies IllegalArgumentException$

updateBoard()

- transition: All non-null cells are moved "down". Finally the remaining `Null` cells are set to different colours.
- output: None
- Exception: None

getCell(i, j)

- transition: None
- output: $out := b[i][j]$
- Exception: $exc := (0 \leq i < SIZE \land 0 \leq j < SIZE) \implies IndexOutOfBoundsException$

2 TODO Controller Module

Abstract Object Module

Controller

Uses

GameState, BoardT, View

Syntax

Exported Constants

Exported Types

Exported Access Program

Routine name	In	Out	Exception
Controller		Controller	
running			
handleBoardClick			
loseFocus			
exit			

Semantics

- State Variables

  board: BoardT

  game: GameState

  win: View

- Environment Variables
- State Invariant
- Assumptions All methods are run after `Controller`.

  `run` is the program main loop. It is called right after `Controller`

  `loseFocus` and `handleBoardClick` are even listeners used by the GUI library.
- Access Routine Semantics
- Local Functions

3 DONE View Module

Abstract Object Module

View

Uses

Syntax

Exported Constants

Exported Types

View = ?

Exported Access Program

Routine name	In	Out	Exception
View			
gameMenu			
startGame	GameState, BoardT		
showPause			
showWin			
showLose			
updateBoard			
connectToMouse	$(\mathbb{N}, \mathbb{N})$, Colour		
drawConnection	$(\mathbb{N}, \mathbb{N})$, $(\mathbb{N}, \mathbb{N})$, Colour		

Semantics

- State Variables

  board: BoardT

  game: GameState

- Environment Variables screen : The screen.

- State Invariant

- Assumptions All functions are called after View has been called.

  startGame is called after gameMenu.

  gamePause is called after startGame.

  showWin and showLose are called only when $state.running() = false$.

  updateBoard is called only after startGame and while $state.running() = true$.

- Access Routine Semantics

  `new` View():

  - transition: screen := Shows a window in the screen with buttons; "Start Game", "Quit".
    - Start Game: Returns "Start Game" message.
    - Quit: Returns "Quit" message.
  - output: out := Selected option.

- Exception: None

gameMenu()

- transition: screen := Shows a menu with a button for each of the available game mode.
- output: The code for the selected game mode.
- Exception: None

startGame(game, b):

- transition: game, board, screen := game, b, shows a grid with coloured dots to represent the board with b[0] being the bottom most row and labels to represent the current game mode/state.
- output: None.
- Exception: None

showPause():

- transition: screen := Shows a menu with "Continue" and "Quit" options.
  - Also calls game.pause() if game is StateTimer
- output: The chosen option.
- Exception: None

showWin():

- transition: screen := Shows a "win" message with buttons "Quit".
- output: The "Quit" message when selected.
- Exception: None

showLose():

- transition: screen := Shows a "lose" message with buttons "Quit".
- output: The "Quit" message when selected.
- Exception: None

updateBoard():

- transition: screen := update the grid to represent the `board`.
- output: None
- Exception: None

drawConnection(coord1, coord2, c):

- transition: screen := Draws a coloured line same colour as `c` from dot at coord1 to dot at coord2.
- output: None
- Exception: None

connectToMouse(coord, c):

- transition: screen := Draw a coloured line same colour as `c` from dot at coord to the current mouse position.
- output: None
- Exception: None

- Local Functions

4 DONE Game State Module

Interface

GameState

Uses

None

Syntax

Exported Constants

None

Exported Types

State = {Win, Lose, Running}

Exported Access Program

Routine name	In	Out	Exception
update			
update	$\mathbb{N}, Colour$		
state		State	
running		$\mathbb{B}$	

Semantics

- State Variables state: State
- Access Routine Semantics

  state()

  - transition: None
  - output: $out := state$
  - Exception: None

  running()

  - transition: None
  - output: $out := state = Running$
  - Exception: None

5 DONE State Timer Module

Template Module inherits GameState

StateTime

Uses

GameState, BoardT

Syntax

Exported Constants

None

Exported Types

StateTime = ?

Exported Access Routine

Routine name	In	Out	Exception
StateTime	$\mathbb{N}$, $\mathbb{N}$	StateTime	
getScore		$\mathbb{N}$	
getRemTime		$\mathbb{R}$	
pause			
unPause			
isPaused		$\mathbb{B}$	

Semantics

- State Variables

  $endTime : \mathbb{R}$

  $curTime : \mathbb{R}$

  $score : \mathbb{N}$

  $scoreGoal : \mathbb{N}$

  $paused : \mathbb{B}$

- Environment Variables

  sysTime: The current system time in seconds.

- State Invariant

  $endTime > 0$

- Assumption

  StateTime is called before any other method is called.

- Access Routine Semantics

  `new` StateTime(i, t):

  - transition: $paused, state, curTime, endTime, score, scoreGoal := false, Running, sysTime, sysTim$ $t, 0, i$

- output: $out := self$
- Exception: None

update()

- transition: $curTime, state := sysTime, (endTime > curTime \implies Running | endTime \leq curTime \implies (score < scoreGoal \implies Lose | True \implies Win))$
- output: None
- Exception: None

update(n, c)

- transition: $paused, score, curTime, state := False, score + calcScore(n, c), sysTime, (endTime > curTime \implies Running | endTime \leq curTime \implies (score < scoreGoal \implies Lose | True \implies Win))$
- output: None
- Exception: None

getScore()

- transition:
- output: $out := score$
- Exception: None

getRemTime()

- transition:
- output: $out := endTime - curTime$
- Exception: None

unPause()

- transition: $paused := False$
- output: None
- Exception: None

pause()

- transition: $paused := True$
- output: None
- Exception: None

isPaused()

- transition: None
- output: $out := paused$
- Exception: None

- Local Functions

calcScore: $\mathbb{N} \times Colour \to \mathbb{N}$
$calcScoret =?$

6 DONE State Score Module

Template Module Inherits GameState

StateScore

Uses

GameState

Syntax

Exported Constants

None

Exported Types

StateScore = ?

Exported Access Routine

Routine name	In	Out	Exception
StateScore	$\mathbb{N}$	StateScore	
getScore		$\mathbb{N}$	
getMaxScore		$\mathbb{N}$	

Semantics

- State Variables

  $score : \mathbb{N}$

  $scoreGoal : \mathbb{N}$

- State Invariant

  $0 \leq score \leq scoreGoal$

- Assumptions

  StateScore is called before any other routine.

- Access Routine Semantics

  `new` StateScore(i)

  - transition: $state, score, scoreGoal := Running, 0, i$
  - output: $out := self$
  - Exception None

  update()

  - transition: None
  - output: None
  - Exception: None

  update(n, c)

- transition: $score, state := score + calcScore(n, c), (score \geq scoreGoal \implies Win | True \implies Running)$
- output: None
- Exception: None

getScore()

- transition:
- output: $out := score$
- Exception: None

getMaxScore()

- transition:
- output: $out := scoreGoal$
- Exception: None

- Local Functions

  calcScore: $\mathbb{N} \times Colour \to \mathbb{N}$
  $calcScore t = ?$

7 DONE State Moves Module

Template Module Inherits GameState

StateMoves

Uses

GameState

Syntax

Exported Constants

Exported Types

StateMoves

Exported Access Routine

Routine name	In	Out	Exception
StateMoves	$\mathbb{N}$, $\mathbb{N}$	StateMoves	
getScore		$\mathbb{N}$	
getScoreGoal		$\mathbb{N}$	
getRemMoves		$\mathbb{N}$	

Semantics

- State Variables

  $score := \mathbb{N}$

  $scoreGoal := \mathbb{N}$

  $moves := \mathbb{N}$

  $maxMoves := \mathbb{N}$

- State Invariant

  $0 \leq score \leq scoreGoal$

  $0 \leq moves \leq maxMoves$

- Assumptions

  StateMoves is called before any other routine.

- Access Routine Semantics

  `new` StateMoves(i, m):

  - transition: $score, scoreGoal, moves, maxMoves := 0, i, 0, m$
  - output: $out := self$
  - Exception: None

  update():

  - transition: None
  - output: None
  - Exception: None

update(n, c)

- transition: $moves, score, state := moves + 1, score + calcScore(n, c), (maxMoves > moves \implies Running | maxMoves \leq moves \implies (score \geq scoreGoal \implies Win | True \implies Lose))$

getScore()

- transition:
- output: $out := score$
- Exception: None

getScoreGoal()

- transition:
- output: $out := scoreGoal$
- Exception: None

getRemMoves()

- transition:
- output: $out := maxMoves - moves$
- Exception: None

- Local Functions

  calcScore: $\mathbb{N} \times Colour) \to \mathbb{N}$
  $calcScore t =?$