Brian Jones

Robert Jong

Alexey Kourganov

Bryant Barron

Group 3, Lab Project 6 (Begins on next page)

2014-02-11

# 

Main Menu

1. Scenario: The Player makes a selection from the Main Menu
2. Triggering Event: Initial Application Screen
3. Actors: Player
4. Related Use Cases:
   1. New Game
   2. Exit Game
5. Stakeholders: Initialize Game, Initialize View, Play Game
6. Pre-Conditions: None
7. Post-Conditions: None
8. Flow of Events:
   1. Player starts App
   2. System displays Main Menu
   3. Player selects from the Menu
   4. System either starts new game or Exits.
9. Exception: None

New Game

1. Scenario: Application begins creating a New Game
2. Triggering Event: Player selects New Game from the Main Menu
3. Actors: Player
4. Related Use Cases:
   1. Initialize Game
   2. Initialize View
   3. Play Game
5. Stakeholders:
   1. Initialize Game
   2. Play Game
   3. Initialize View
6. Pre-Conditions: None
7. Post-Conditions: Game is now to be created
8. Flow of Events:
   1. Player selects New Game from the Main Menu
   2. System begins processing new game
9. Exception: None

Exit Game

1. Scenario: System will now close and destroy the Application
2. Triggering Event: Player selects Exit Game from the Main Menu
3. Actors: Player
4. Related Use Cases: None
5. Stakeholders: None
6. Pre-Conditions: None
7. Post-Conditions: Game is now to be destroyed
8. Flow of Events:
   1. Player selects Exit Game from the Main Menu
   2. System begins closing the Application
9. Exception: None

Initialize Game

1. Scenario: Initialize all variables and objects relevant to the Game framework and algorithms
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases:
   1. Initialize Timer
   2. Initialize Score
   3. Initialize Terrain
   4. Initialize Input Devices
   5. Create Obstacles and Power-ups
   6. Create Snake
5. Stakeholders:
   1. Initialize View
   2. Play Game
6. Pre-Conditions: None
7. Post-Conditions:
   1. Game Framework Initialized
8. Flow of Events:
   1. Initialize Timer
   2. Initialize Score
   3. Initialize Input Devices
   4. Create Obstacles and Power-Ups
   5. Create Snake
9. Exception:
   1. C-2: Exit Game if no acceptable Input Devices exist

Play Game

1. Scenario: Begins the actual gaming algorithms
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases:
   1. Move Snake
   2. Grow Snake
   3. Turn Snake
   4. End Game
5. Stakeholders: End Game
6. Pre-Conditions:
   1. Game has been Initialized
   2. View has been initialized
7. Post-Conditions: None
8. Flow of Events:
   1. Start the Timer
   2. Begin the Gaming algorithms and functions in a loop until the game ends
9. Exception: None

Initialize View

1. Scenario: The Visual aspects of the Application will be initialized here
2. Triggering Event: Player Selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases:
   1. Draw Snake
   2. Draw Obstacles and Power-Ups
   3. Clear Screen
   4. Draw Terrain
5. Stakeholders:
   1. Play Game
6. Pre-Conditions:
   1. Game Framework Initialized
7. Post-Conditions: Game View Initialized
8. Flow of Events:
   1. Clear Screen
   2. Draw Terrain
   3. Draw Obstacles and Power-Ups
   4. Draw Snake
9. Exception: None

Move Snake

1. Scenario: Moves the Snake to a new position
2. Triggering Event: Timer
3. Actors: Automated/Timer
4. Related Use Cases: Collision Check
5. Stakeholders: None
6. Pre-Conditions: Collision Checker cannot return Out-of-Bounds or Cannibal
7. Post-Conditions: Snake has moved
8. Flow of Events:
   1. Check for Collisions with the Boundary or with itself (Cannibal)
   2. If no Collisions, then Move Snake
9. Exception: None

Grow Snake

1. Scenario: The Snake needs to grow
2. Triggering Event: The Snake has eaten a power-up
3. Actors: Player
4. Related Use Cases: Collision Check
5. Stakeholders: None
6. Pre-Conditions: Collision Check returns Power Up status
7. Post-Conditions: Snake is longer
8. Flow of Events:
   1. Check for Collision with a Power-Up
   2. If Collision is present, then make the snake get longer
9. Exception: None

Turn Snake

1. Scenario: The Snake needs to turn
2. Triggering Event: The Player has moved the Device
3. Actors: Input Sensors (Accelerometer, Gyroscope, and Gravity Sensors)
4. Related Use Cases: Device Movement
5. Stakeholders: None
6. Pre-Conditions: The Player has moved the Device in an amount exceeding a preset threshold
7. Post-Conditions: None
8. Flow of Events:
   1. Check for Device Movement
   2. If Device has moved, check to see if the Device has moved past a predefined threshold
   3. If the Device has moved significantly, register that movement
   4. Now turn the snake in the direction that corresponds with the Direction of the Device’s movement
9. Exception: None

End Game

1. Scenario: The Game is Over
2. Triggering Event: The Snake has either eaten itself or run into the outer boundary of the terrain
3. Actors: Player
4. Related Use Cases:
   1. Exit Game
   2. Collision Check
5. Stakeholders: Main Menu
6. Pre-Conditions: Collision Check has returned either Out-of-Bounds or Cannibal
7. Post-Conditions: Ready for Main Menu
8. Flow of Events:
   1. Check for either Out-of-Bounds or Cannibal Collisions
   2. If detected, then the game is over
9. Exception: None

Initialize Timer

1. Scenario: Initializes the Timer
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: None
5. Stakeholders: Play Game
6. Pre-Conditions: None
7. Post-Conditions: None
8. Flow of Events:
   1. Initialize the Timer to zero
   2. Synchronize the Timer with the Time
9. Exception: None

Initialize Score

1. Scenario: Initializes the Score
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: None
5. Stakeholders: Play Game
6. Pre-Conditions: None
7. Post-Conditions: None
8. Flow of Events:
   1. Initialize Score to Zero
9. Exception: None

Initialize Terrain

1. Scenario: Initializes the Terrain
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: Draw Terrain
5. Stakeholders: Draw Terrain
6. Pre-Conditions: None
7. Post-Conditions: None
8. Flow of Events:
   1. Initialize the Terrain from map files
9. Exception: None

Initialize Input Devices

1. Scenario: Sets up the Input Sensors for use by the game
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: None
5. Stakeholders: Turn Snake
6. Pre-Conditions: Accelerometer, Gyroscope, and Gravity Sensors must exist
7. Post-Conditions: None
8. Flow of Events:
   1. Create Handles to the Sensors
   2. Create Event Listener for the Sensor Handles
9. Exception: None

Create Obstacles and Powerups

1. Scenario: Creates the Terrain Obstacles and Powerups (and their locations)
2. Triggering Event: Player Selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: Draw Obstacles and Powerups
5. Stakeholders: Draw Obstacles and Powerups
6. Pre-Conditions: Terrain must have been properly Initialized
7. Post-Conditions: None
8. Flow of Events:
   1. Determine Terrain size
   2. Use the Terrain size to determine Obstacle sizes and Positions
   3. Use the Terrain size and Obstacle sizes and Positions to determine Power-up Locations
   4. Create Obstacles and Power-Ups
9. Exception: None

Create Snake

1. Scenario: Creates the Snake object
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: Draw Snake
5. Stakeholders: Draw Snake
6. Pre-Conditions: Terrain must have been properly Initialized
7. Post-Conditions: None
8. Flow of Events:
   1. Use the Terrain to determine an acceptable location for the snake to be placed
   2. Place the Snake in that location
9. Exception: None

Draw Snake

1. Scenario: Draws the Original Snake, as well as updating Snake size and position
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases:
   1. Create Snake
   2. Move Snake
   3. Grow Snake
5. Stakeholders:
   1. Move Snake
   2. Grow Snake
   3. Collision Check
6. Pre-Conditions: Snake must have been created properly
7. Post-Conditions: None
8. Flow of Events:
   1. Check position of Snake
   2. Move to the Snake’s tail and get that element’s size/position
   3. Draw the tail
   4. Move towards the head by one element, and get that element’s size/position
   5. Draw that element
   6. Repeat until there are no further elements to access (meaning the Head has been drawn)
9. Exception: None

Draw Obstacles and Powerups

1. Scenario: Draws the Obstacles and Power-Ups on the Terrain
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: Create Obstacles and Powerups
5. Stakeholders: None
6. Pre-Conditions: Obstacles and Powerups must have been properly initialized
7. Post-Conditions: None
8. Flow of Events:
   1. Read size/position of Obstacles
   2. Draw Obstacles
   3. Read size/position of Powerups
   4. Draw Powerups
9. Exception: None

Clear Screen

1. Scenario: Clears the Screen for new Views to be displayed
2. Triggering Event: Player Selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: None
5. Stakeholders:
   1. Initialize Terrain
   2. Create Snake
   3. Create Obstacles and Powerups
6. Pre-Conditions: None
7. Post-Conditions: Screen must be clear
8. Flow of Events:
   1. Erase everything on the screen
9. Exception: None

Draw Terrain

1. Scenario: Draws the current game’s terrain to the screen
2. Triggering Event: Player selected New Game from the Main Menu
3. Actors: Player
4. Related Use Cases: Initialize Terrain
5. Stakeholders: None
6. Pre-Conditions: Terrain must have been properly Initialized
7. Post-Conditions: Terrain displayed to screen
8. Flow of Events:
   1. Read Terrain object
   2. Display Terrain object
9. Exception: None

Collision Check

1. Scenario: Checks current snake position for Collisions
2. Triggering Event: Move Snake
3. Actors: Timer/Automated
4. Related Use Cases:
   1. Power Up
   2. Out-of-Bounds
   3. Cannibal
   4. No Event
5. Stakeholders:
   1. Move Snake
6. Pre-Conditions: None
7. Post-Conditions: Any collisions have been detected
8. Flow of Events:
   1. Check if Power Up returns true; If so, return Power Up
   2. Check if Out-of-Bounds returns true; If so, return Out-of-Bounds
   3. Check if Cannibal returns true; If so, return Cannibal
   4. If none of the above returns true, return No Event
9. Exception: None

Power Up

1. Scenario: Determines if a Power Up has been eaten
2. Triggering Event: Snake Moved
3. Actors: Player
4. Related Use Cases:
   1. Collision Check
5. Stakeholders:
   1. Collision Check
   2. Move Snake
   3. Grow Snake
   4. End Game
6. Pre-Conditions: Snake must have moved
7. Post-Conditions: None
8. Flow of Events:
   1. Check if current position of snake’s head is on top of a power-up
   2. If so, return true
   3. Else, return false
9. Exception: None

Out-of-Bounds

1. Scenario: Determines if snake has moved out of the playable Terrain
2. Triggering Event: Snake moved
3. Actors: Player
4. Related Use Cases:
   1. Collision Check
5. Stakeholders:
   1. Collision Check
   2. Move Snake
   3. Grow Snake
6. Pre-Conditions: Snake must have moved
7. Post-Conditions: None
8. Flow of Events:
   1. Check if snake’s head is currently outside the playable arena.
   2. If yes, return true
   3. Else, return false
9. Exception: None

Cannibal

1. Scenario: Checks to see if the snake has eaten itself
2. Triggering Event: Snake has moved
3. Actors: Player
4. Related Use Cases:
   1. Collision Check
5. Stakeholders:
   1. Collision Check
   2. Move Snake
   3. Grow Snake
6. Pre-Conditions: Snake must have moved
7. Post-Conditions: None
8. Flow of Events:
   1. Check if snake’s head is currently occupying the same position as an element of its body
   2. If yes, return true
   3. Else, return false
9. Exception: None

No Event

1. Scenario: Checks to see if the Snake is simply occupying innocuous space
2. Triggering Event: Snake has moved
3. Actors: Player
4. Related Use Cases:
   1. Collision Check
5. Stakeholders:
   1. Collision Check
   2. Move Snake
   3. Grow Snake
6. Pre-Conditions: Snake must have moved
7. Post-Conditions: None
8. Flow of Events:
   1. Check to see if Power-Up is true; If yes, return false
   2. Check to see if Out-of-Bounds is true; If yes, return false
   3. Check to see if Cannibal is true; If yes, return false
   4. Else, return true
9. Exception: None

Device Movement

1. Scenario: Reports that the Device has significantly moved
2. Triggering Event: The Player moved the Device
3. Actors: Player
4. Related Use Cases: Turn Snake
5. Stakeholders: Turn Snake
6. Pre-Conditions: Device must possess Accelerometer, Gyroscope, and Gravity Sensors
7. Post-Conditions: Measurements of Device’s Input Sensors have been submitted
8. Flow of Events:
   1. Listen for Accelerometer Sensor movement
   2. Listen for Gyroscope Sensor movement
   3. Listen for Gravity Sensor movement
   4. Report Incorporated Matrix vector to Listening object
9. Exception: If no Accelerometer, Gyroscope, and/or Gravity Sensors exist, exit game