Project Part II

Yun Zhou

Project Summary

An advanced snake game is to be developed in android platform. People can use their cell phone to play this small game at their spare time and get relaxed.
The game is played by single player, whose goal is to obtain the highest score by eating good food. Before the game begins, the player need to enter the username, the player can choose easy/medium/hard modes of games. The player can also modify the game interface. Once the game begins, the player controls the movement of the snake. At any time, the snake can move in one of the 4 directions which the player can control through touching the screen. When the snake eats food, its speed and scores will change. If the snake eats good food, its speed will decrease and the player will win some scores; if the snake eats bad food, its speed will increase and the player will lose some scores. When the snake hit itself, the obstacles or the boundary, it will die and the game is over. When the game is over, the player can choose play again or exit.

Requirements

User Requirements				
ID	Description	Topic Area	User	Priority
UR-01	As a client, I want the system asks for player to enter a unique username before playing game so that the database can retrieve the player’s username and related score.	Entering Username	Player	Critical
UR-02	As a client, I want the system provides start game button so that the player be able to begin game in an easy process (no login required).	Playing Game	Player	High
UR-03	As a player, I want the system provides setting button so that I am able to adjust the color, difficulty mode and volume.	Modifying Setting	Player	Medium
UR-04	As a player, I can view top ten scores by clicking view score button on the main menu.	View Score	Player	Medium

UR-05	As a player, I can use pause/resume button during playing the game so that I can take a break and play again.	Pause Game	Player	Low
UR-06	As a player, I want to be provided an exit button so that I can exit the game. I want to be able to do this either after the game is over or even while playing the game.	Exit Game	Player	High

Functional Requirements				
ID	Description	Topic Area	User	Priority
FR-01	The direction of the snake will be controlled by touching screen (up, down, left and right).	Control Game	Player	High
FR-02	The game will be displayed on visual display of specific size.	Display	System	High
FR-03	The game will have a main menu to show help, setting, view score, play game and exit buttons.	Title Screen	Player	High
FR-04	A message “game over” pops up when the game over.	Game Over	System	High
FR-05	Scoring in game will be based on the amount of good food eaten by the snake.	Score	System	High
FR-06	The current score will be displayed on the screen when player is playing the game.	Score	System	High
FR-07	The game will have background music when the player is playing the game.	Sound	System	Medium

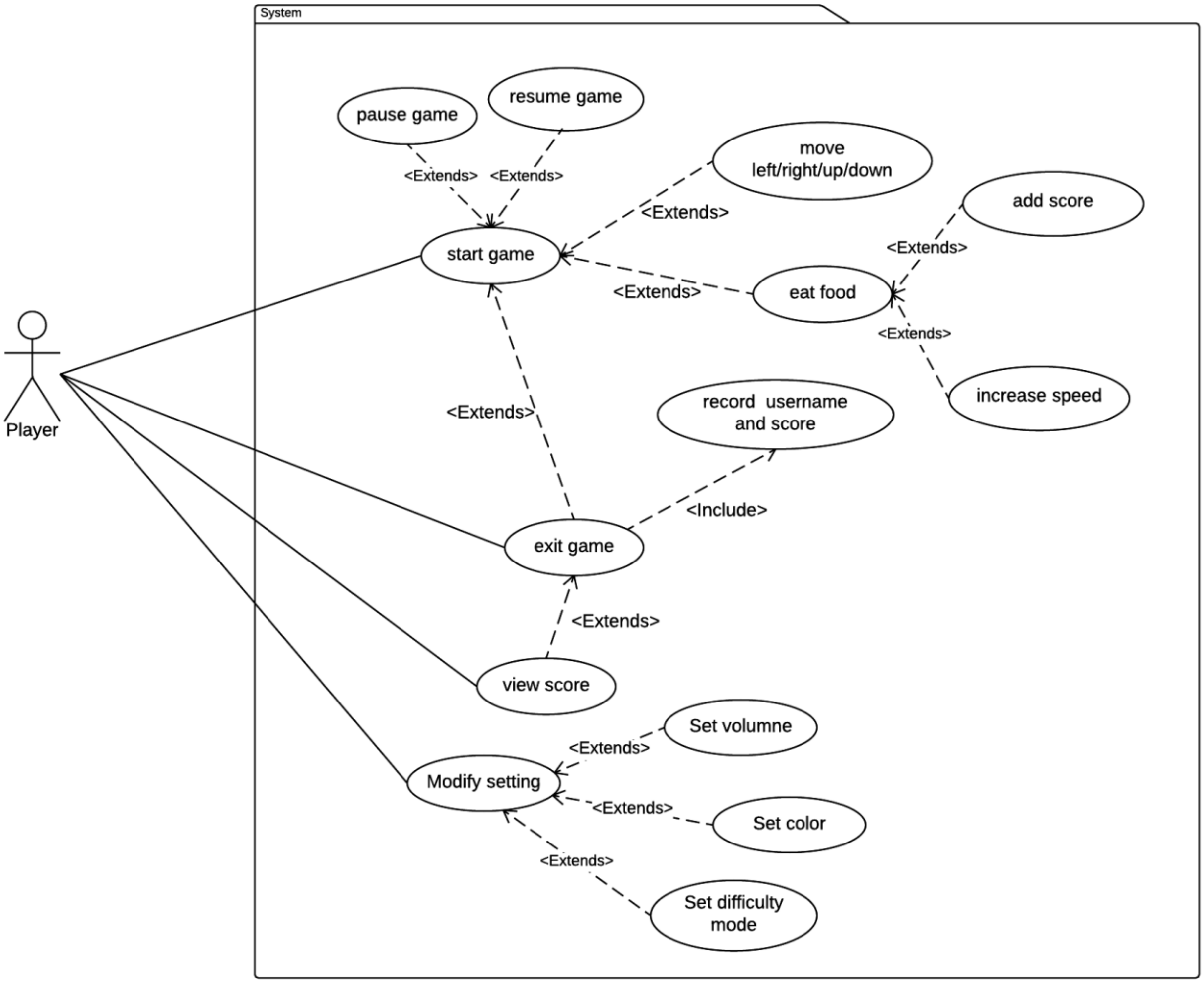
Non-Functional Requirements

ID	Description	Topic Area	User	Priority
NFR-01	Performance : 1) The game loading time should be under 10s.	System’s Functionality	Player System	High
NFR-02	Performance : 2) All pages should be displayed under 3s.	System’s Functionality	Player System	High
NFR-03	Performance : 3) It only supports one player in the game.	System’s Functionality	Player System	High
NFR-04	Performance : 4) The requirement of resource for data storage is medium.	System’s Functionality	Player System	High
NFR-05	Platform Constraints: The game will be installed just on android system.	Platform	Player	Medium
NFR-06	Usability: The game interface should be designed to be familiar for the player. It should have similar design with classical snake game.	Interface	All	High
NFR-07	Supportability: In future it can be extended to other different platforms.	Support	Player	Low
NFR-08	Interface: This system will interact with a database system which stores the players’ usernames and scores.	Intercation	System	Medium

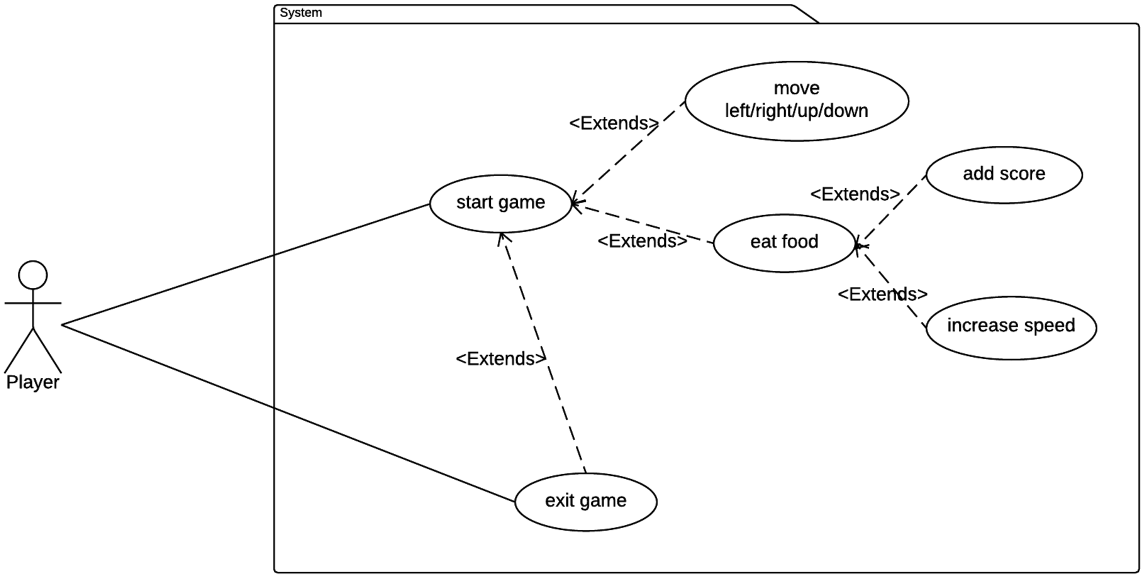
Business Requirements				
ID	Description	Topic area	User	Priority
BR-01	Players can share their names and scores through Facebook or Twitter.	Social Network	Player	Low

User and Tasks

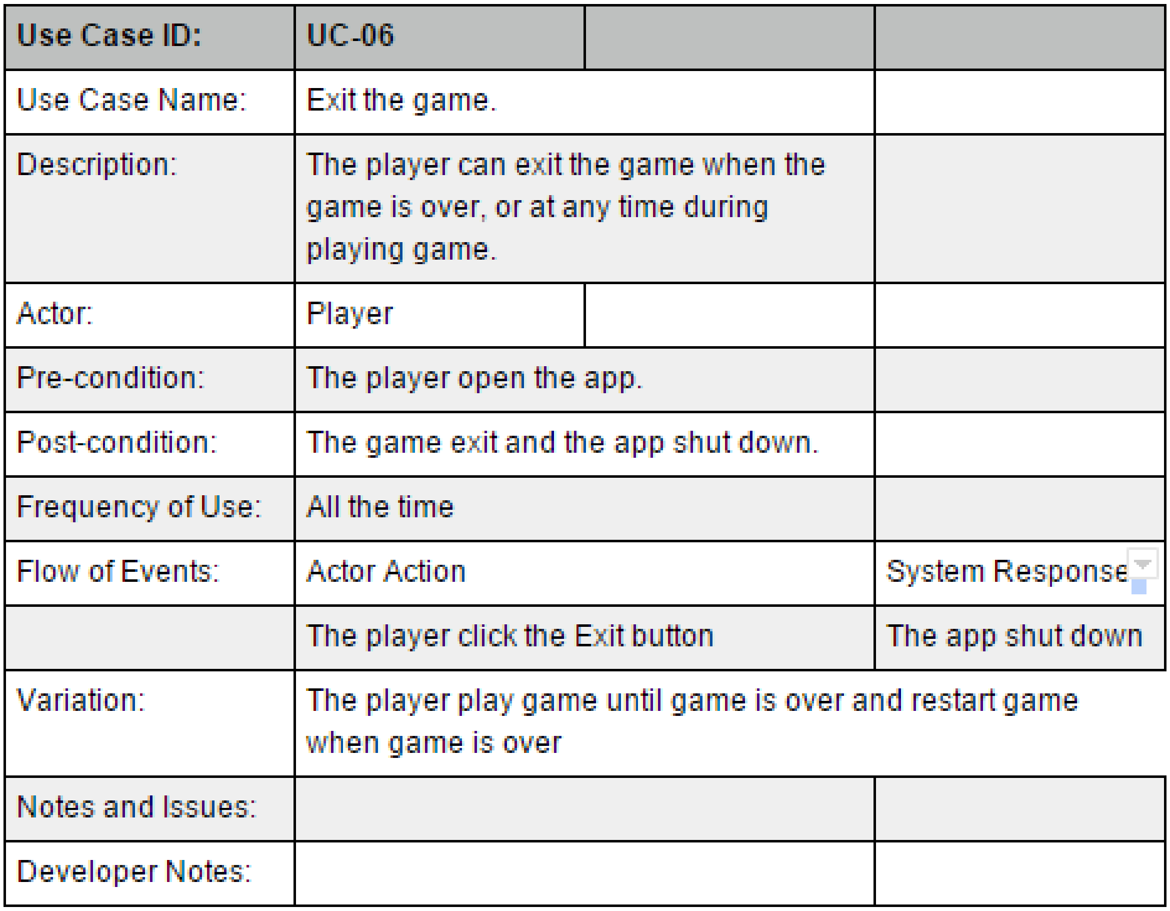
The whole use case diagram:



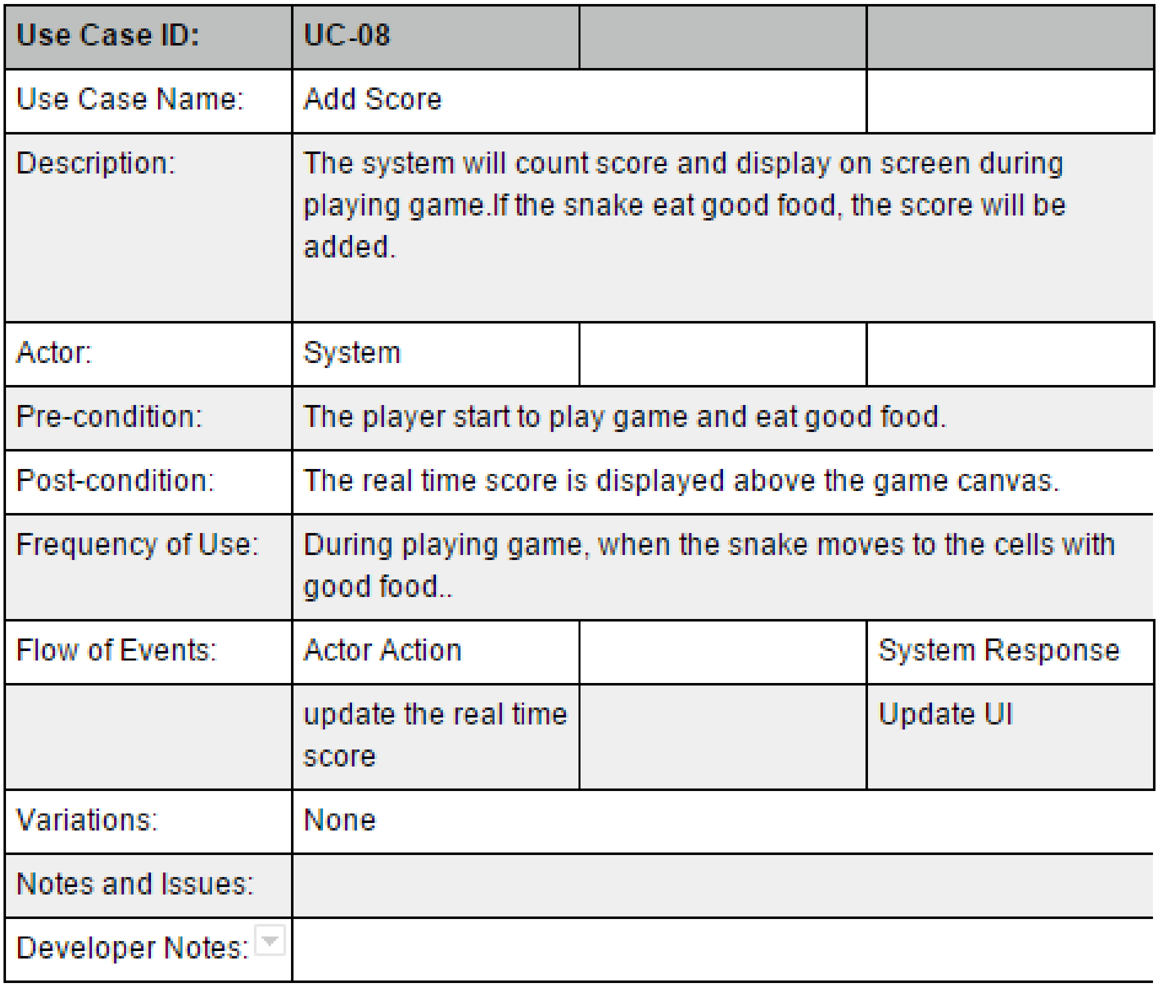
The “start game” use case and related use cases diagram, which I focus on:



Use Case:	**UC-03**				
Use Case Name:	Start Game				
Description:	The player is in control of a snake moving around the a square field of cells to four directions. The player is in control of a snake to eat different kinds of food.				
Actors:	Player				
Pre-conditions:	The player start to play game. In the game canvas, the system draw a square field of cells and a snake with a number of cells.				
Post-conditions:	The snake move four direction: left, right, up and down. The player will win scores when the snake eats good food; The speed of the snake will increase if it eats bad food.				
Frequency of Use:	During playing				
Flow of Events:	Actor Action			System Response	
	1. Touch the screen to move left. 2. Touch the screen to move right. 3. Touch the screen to move up. 4. Touch the screen to move down. 5. Move to the food position and the food is good. 6. Move to the food position and the food is bad.			1. Snake moves to the left; 2. Snake moves to the right; 3. Snake moves up; 4. Snake moves down; 5. Scores is added; 6. Snake speed increases.	
Variations:	The player pause the game, turn off sound, and exit game.				
Notes and Issues:					
Developer Notes:					



Use Case ID:	**UC-06**		
Use Case Name:	Exit the game.		
Description:	The player can exit the game when the game is over, or at any time during playing game.		
Actor:	Player		
Pre-condition:	The player open the app.		
Post-condition:	The game exit and the app shut down.		
Frequency of Use:	All the time		
Flow of Events:	Actor Action		System Response
	The player click the Exit button		The app shut down
Variation:	The player play game until game is over and restart game when game is over		
Notes and Issues:			
Developer Notes:			

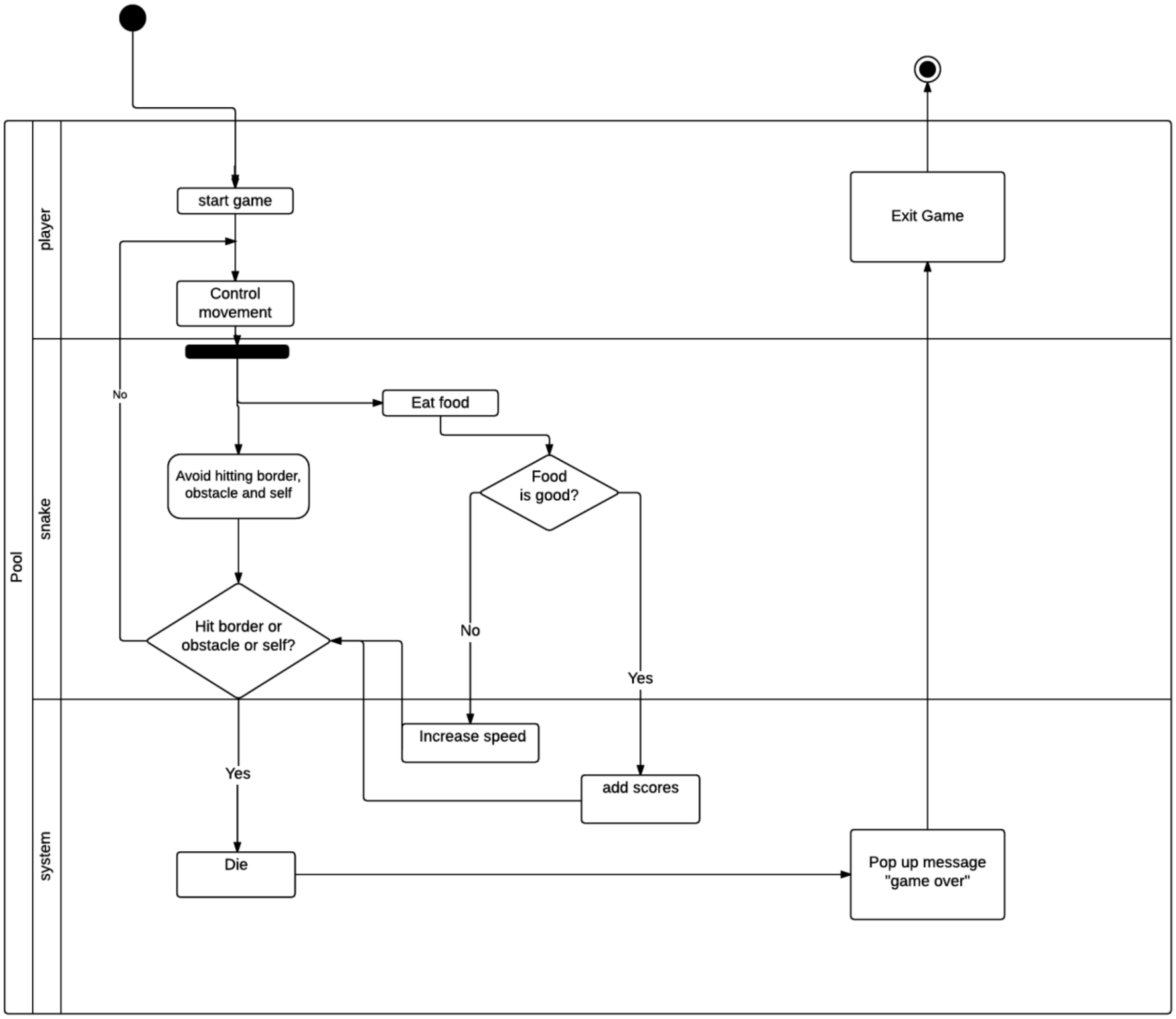


Use Case ID:	**UC-08**		
Use Case Name:	Add Score		
Description:	The system will count score and display on screen during playing game.If the snake eat good food, the score will be added.		
Actor:	System		
Pre-condition:	The player start to play game and eat good food.		
Post-condition:	The real time score is displayed above the game canvas.		
Frequency of Use:	During playing game, when the snake moves to the cells with good food..		
Flow of Events:	Actor Action		System Response
	update the real time score		Update UI
Variations:	None		
Notes and Issues:			
Developer Notes:			

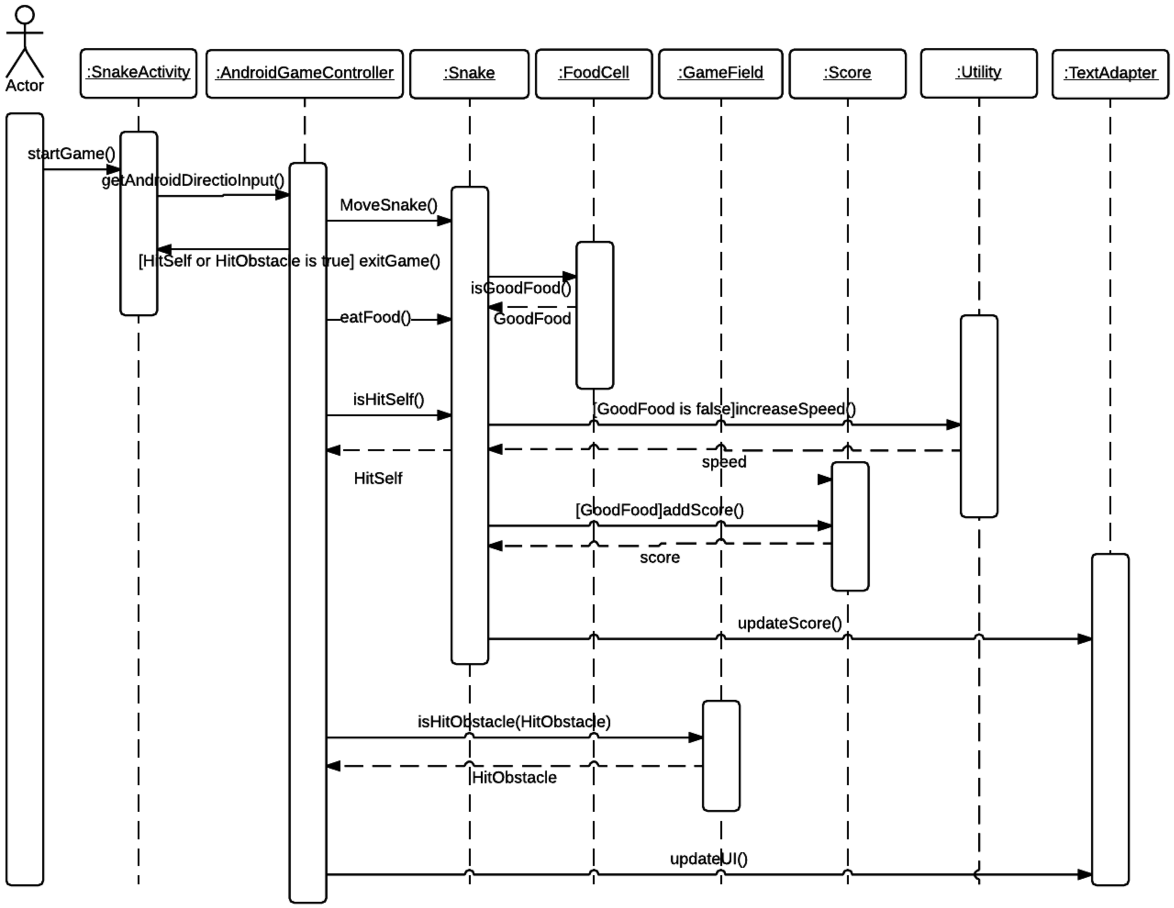
Use Case ID:	**UC-09**		
Use Case Name:	Increase Speed		
Description:	The system will modify the speed of the snake movement. If the snake eat bad food, its speed will increase.		
Actor:	System		
Pre-condition:	The player start to play game and eat bad food.		
Post-condition:	The snake will move faster.		
Frequency of Use:	During playing game, when the snake moves to the cells with bad food..		
Flow of Events:	Actor Action		System Response
	update the real time snake speed		Update UI
Variations:	None		
Notes and Issues:			
Developer Notes:			

Activity Diagram

The activity diagram focuses on the activities occurring from the player starting game to the game over. In this activity diagram, I design that right after popping up message “game over”, the user exits game. Because another group member will focus on the view score, exit and some other use cases, I need to leave her some room to do so.

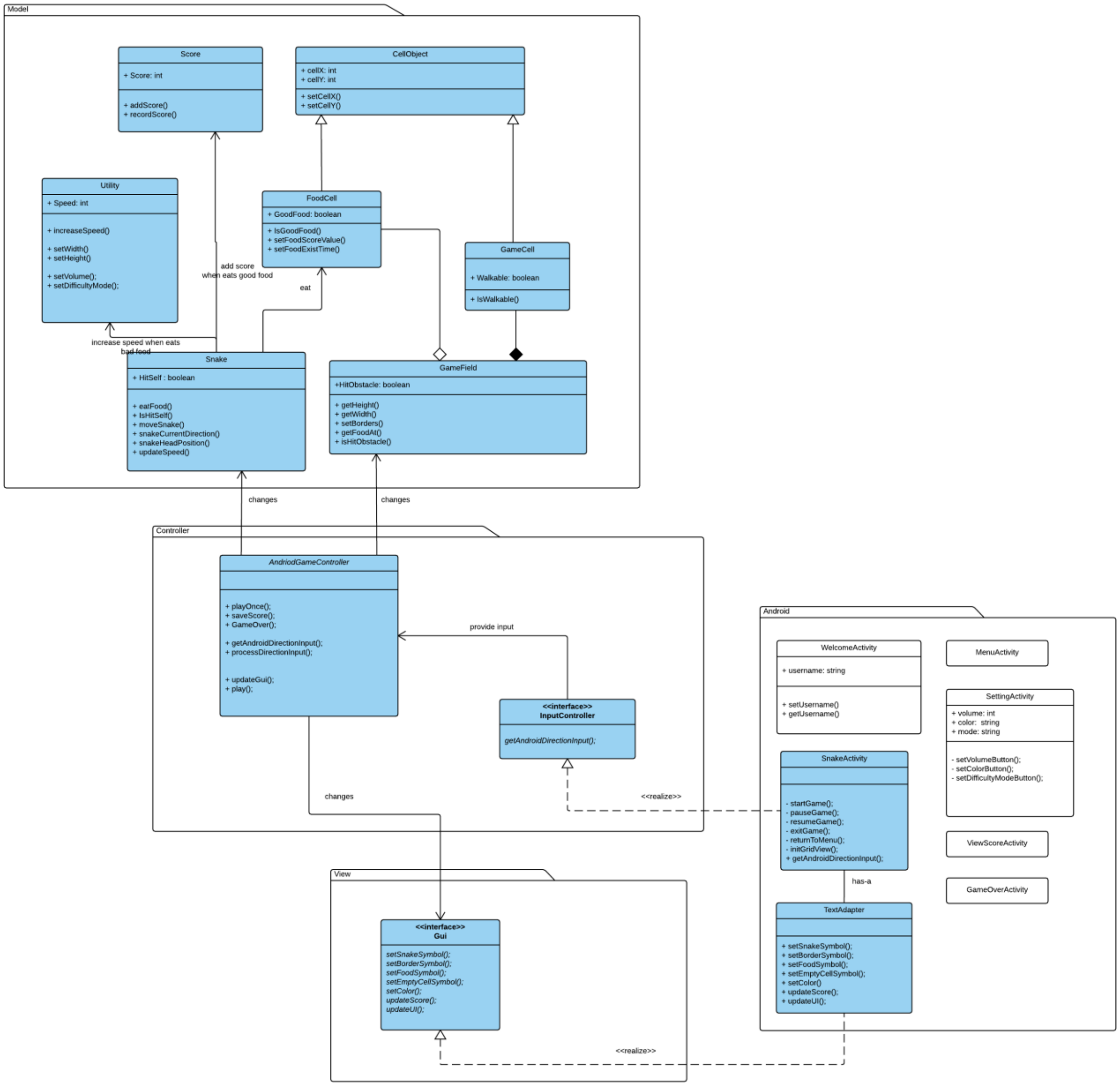


Sequence diagram



Class Diagram

Classes with blue background are the classes associated with the use cases I focus on.


Model
Score
+ Score: int
+ addScore()
+ recordScore()
CellObject
+ cellX: int
+ cellY: int
+ setCellX()
+ setCellY()
Utility
+ Speed: int
+ increaseSpeed()
+ setWidth()
+ setHeight()
+ setVolume();
+ setDifficultyMode();
FoodCell
+ GoodFood: boolean
+ IsGoodFood()
+ setFoodScoreValue()
+ setFoodExistTime()
GameCell
+ Walkable: boolean
+ IsWalkable()
add score
when eats good food
eat
increase speed when eats
bad food
Snake
+ HitSelf : boolean
+ eatFood()
+ IsHitSelf()
+ moveSnake()
+ snakeCurrentDirection()
+ snakeHeadPosition()
+ updateSpeed()
GameField
+HitObstacle: boolean
+ getHeight()
+ getWidth()
+ setBorders()
+ getFoodAt()
+ isHitObstacle()
changes
changes
Controller
AndriodGameController
+ playOnce();
+ saveScore();
+ GameOver();
+ getAndroidDirectionInput();
+ processDirectionInput();
+ updateGui();
+ play();
provide input
<<interface>>
InputController
getAndroidDirectionInput();
changes
<<realize>>
Android
WelcomeActivity
+ username: string
+ setUsername()
+ getUsername()
MenuActivity
SettingActivity
+ volume: int
+ color: string
+ mode: string
- setVolumeButton();
- setColorButton();
- setDifficultyModeButton();
SnakeActivity
- startGame();
- pauseGame();
- resumeGame();
- exitGame();
- returnToMenu();
- initGridView();
+ getAndroidDirectionInput();
ViewScoreActivity
has-a
GameOverActivity
TextAdapter
+ setSnakeSymbol();
+ setBorderSymbol();
+ setFoodSymbol();
+ setEmptyCellSymbol();
+ setColor()
+ updateScore();
+ updateUI();
View
<<interface>>
Gui
setSnakeSymbol();
setBorderSymbol();
setFoodSymbol();
setEmptyCellSymbol();
setColor();
updateScore();
updateUI();
<<realize>>