Project Part 2

Team: Brandon Barrett, Brandon Jacquez, Isabella Figueroa, Dilara Madinger.
Title: Biking Game.
Project Summary:
A simple 2-D side scroller game with the bike racer jumping over frogs, ducking birds, and trying to part ways safely with an occasional incoming bike.

- User will push 4 arrow keys for going faster, slower, jumping, ducking.
- There is a damage meter.
- There is a scoreboard.
- There is a time meter.
- There is collision detector.
- Audio for collisions, background tune.
- Usage of sprite animation for the racer figure.
- Store high scores.
- Get a name for high score.
- Get options to save or pause the game.

Possible extras:

- Integrate more popular physics library from open source.
- Multi player mode for several racers.

Project Requirements

Business Requirements

We have no Business Requirements for this game.

User Requirements

ID	Requirement	Topic Area	Priority
UR-1	As a user, I need to be able to end a game with the option of saving it.	Gameplay	High
UR-2	As a user I want to pause the game.	User Experience	Low
UR-3	As a user, I want to move my character forward, speed up, slow down, jump, and duck.	Gameplay	Critical
UR-4	As a user I want to have objects on	Gameplay	High

	the track that I can collide with if I don’t jump or duck.		
UR-5	As a user I want to have a way to track the damage I accumulate when I collide with obstacles.	User Experience	Medium
UR-6	As a user I want to have a time tracker.	User Experience	Medium
UR-7	As a user, I want to track my score based on time and damage.	Statistics	Medium
UR-8	As a user I want penalty if my bike accumulates too much damage.	User Experience	Medium
UR-9	As a user I want to be able to view high scores and my score.	Statistics	Medium
UR-10	As a user I want to have background and collision sounds in the game.	User Experience	Low
UR-11	As a user, I want to encounter another moving object that makes random decisions, so that I can react quickly to avoid it.	Gameplay	Medium
UR-12	As a user, I want to put in my nickname, so that I can track my progress.	Statistics	Medium

Functional Requirements

ID	Requirement	Topic Area	Priority
FR-1	When the user opens the game, there will be a dialogue window for them to input their nickname.	User Experience	Medium
FR-2	Once the name of the user is in, there is a start game option.	Functionality	Critical
FR-3	When user selects “High Scores” option from main page, database	User Experience	Medium

	will be queried and display high scores.		
FR-5	The end of the game occurs at the end of the time or with passing the threshold for damage.	Gameplay	High
FR-6	There will be a pause button for the user to pause.	User Experience	Low
FR-7	The scoreboard will be sorted in descending order.	Functionality	Medium

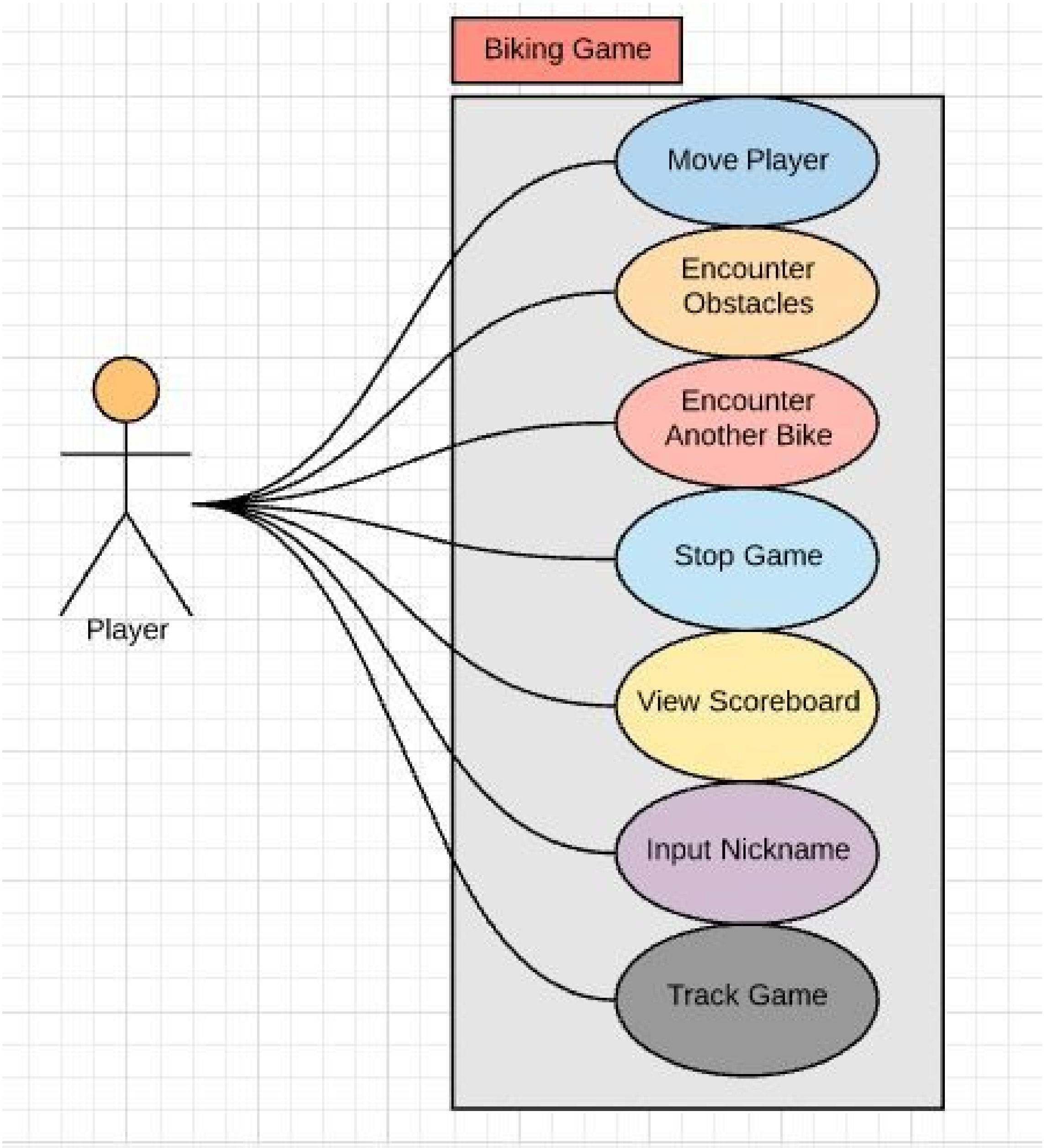
Non-Functional Requirements

ID	Requirement	Topic Area	Priority
NF-1	The user will download the game to play locally.	Access to the Game	Medium
NF-2	As a reference there will be an option to read the rules.	Gameplay	Low
NF-3	The nicknames and scores will be stored in database.	Storage	High
NF-4	The game must run fast.	Performance	Critical

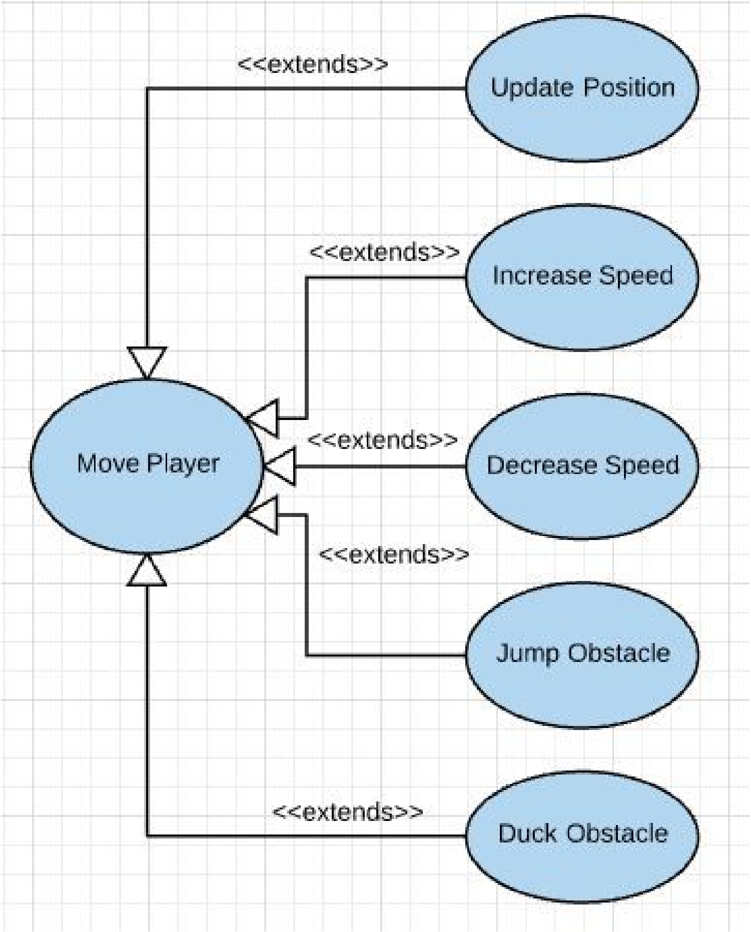
Use Case Diagrams

Actors: Player.

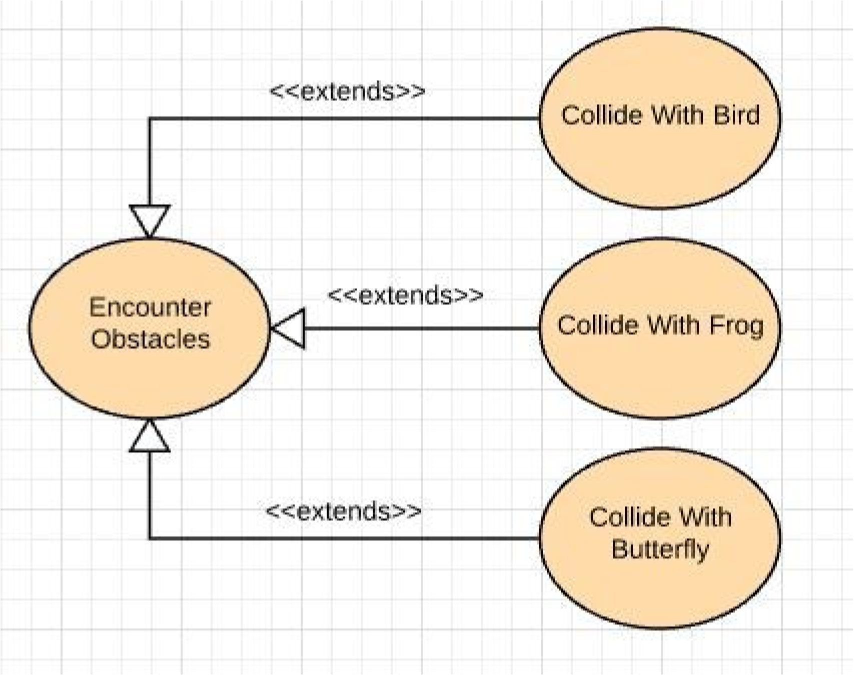
Use Case Overview:

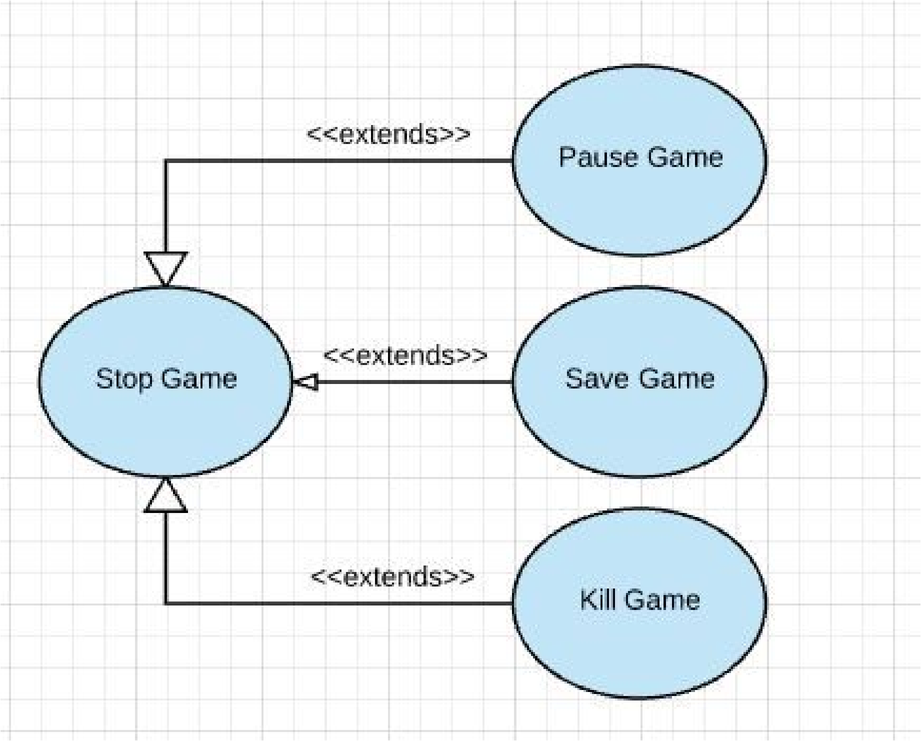


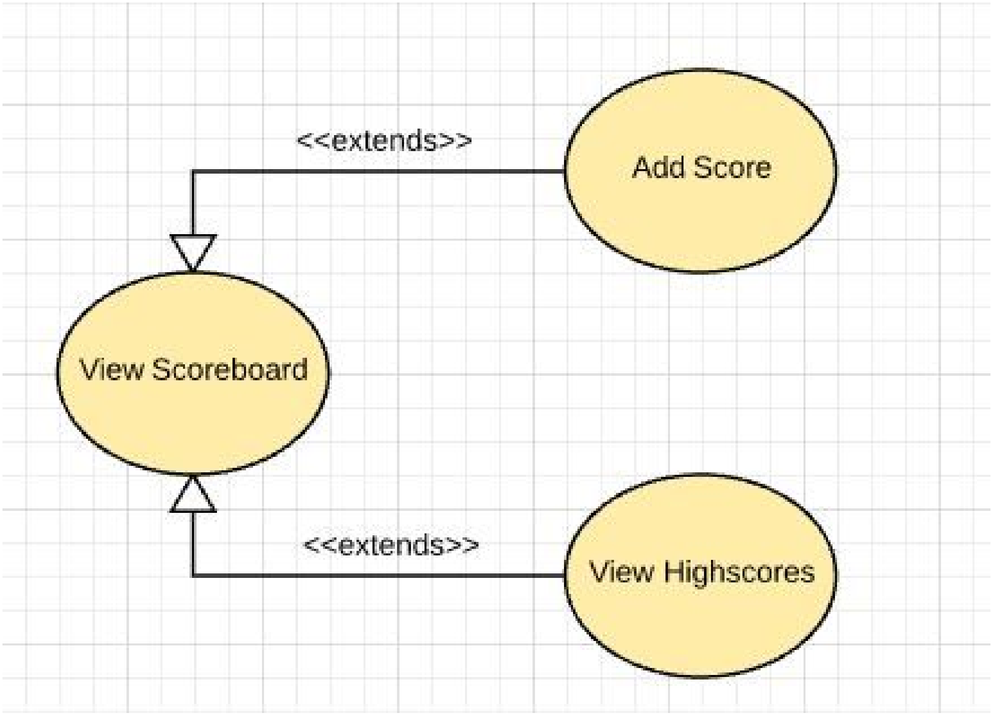
Sub-diagrams:



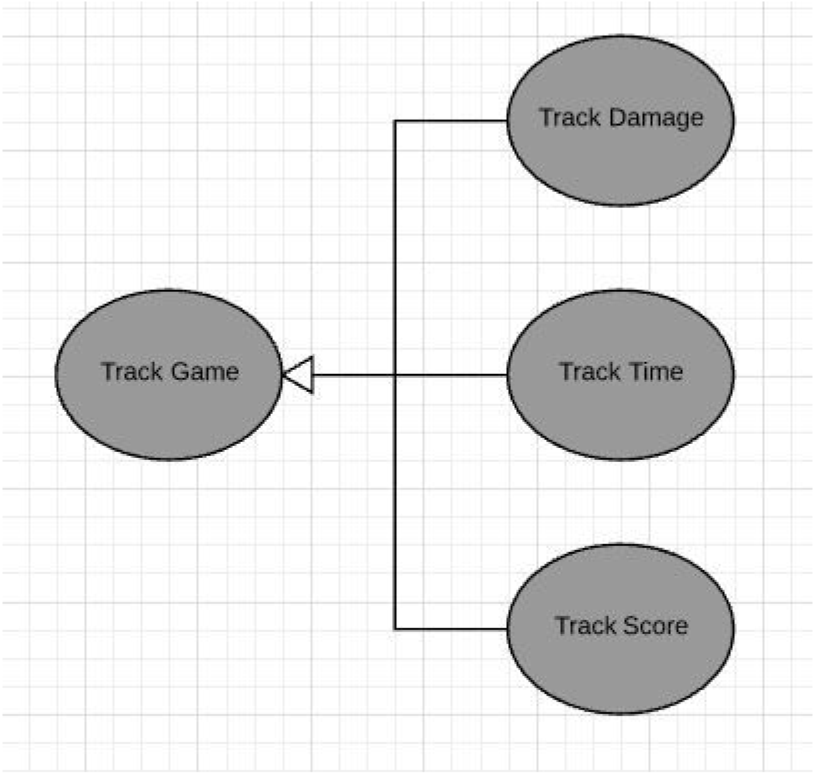
-


<<extends>>
Collide With Bird
<<extends>>
Encounter Obstacles
Collide With Frog
<<extends>>
Collide With Butterfly


<<extends>>
Pause Game
<<extends>>
Stop Game
Save Game
<<extends>>
Kill Game



<<extends>>
Add Score
View Scoreboard
<<extends>>
View Highscores



Track Damage
Track Game
Track Time
Track Score

Use Case Documentation

<table>
<tr><td><b>Use Case ID:</b></td><td>UC-1</td></tr>
<tr><td><b>Use Case Name:</b></td><td>Stop Game</td></tr>
<tr><td><b>Description:</b></td><td>Player has an option to exit the game saving his/her progress or not.</td></tr>
<tr><td><b>Actors:</b></td><td>Player.</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player pressed ‘Esc’ button.</td></tr>
<tr><td><b>Post-conditions:</b></td><td>The current game is closed and is saved in memory, so that player can restart from that stopping point</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Often.</td></tr>
<tr><td><b>Flow of Events:</b></td><td>
<table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player selects an option to save the game or exit without saving.</td><td>If selected not to save the game, clears the memory allocated for the game session.</td></tr>
<tr><td><b>2</b></td><td></td><td>If selected to save the game, the system saves the state of the game.</td></tr>
</table>
</td></tr>
<tr><td><b>Variations:</b></td><td>none</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Brandon Barrett.</td></tr>
</table>

Use Case ID:	UC-2
Use Case Name:	Pause Game.
Description:	Player has the option to pause the game, without need to *save* state, mid-game.

<table>
<tr><td><b>Actors:</b></td><td>Player</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>The player will press the ‘Esc’ key to bring up the “Stop Menu”..</td></tr>
<tr><td><b>Post-conditions:</b></td><td>The current game state is frozen.</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Often.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player presses the ‘Esc” key.</td><td>System freezes game state.</td></tr>
<tr><td><b>2</b></td><td></td><td>System displays “Stop Menu” to user, until player chooses to resume.</td></tr>
</table></td></tr>
<tr><td><b>Variations:</b></td><td>none</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Brandon Barrett.</td></tr>
</table>

<table>
<tr><td><b>Use Case ID:</b></td><td>UC-3</td></tr>
<tr><td><b>Use Case Name:</b></td><td>Move Player.</td></tr>
<tr><td><b>Description:</b></td><td>Player presses one of the four arrow key buttons to progress through bike course.</td></tr>
<tr><td><b>Actors:</b></td><td>Player</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player will press arrow keys, to specify various movements.</td></tr>
<tr><td><b>Post-conditions:</b></td><td>Player’s position will be updated, and speed adjusted in the direction specified by the button they press.</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Constant throughout the game.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td><b>1</b></td><td>Player presses the key.</td><td>System updates position and speed of player.</td></tr>
<tr><td><b>2</b></td><td></td><td>System renders player’s new position on the screen.</td></tr>
</table></td></tr>
<tr><td><b>Variations:</b></td><td>Press Right Arrow Key - Move forward.
Hold Right Arrow Key - Increase speed.
Press Left Arrow Key - Slow down
Press Up Arrow Key - Jumping motion.
Press Down Arrow Key - Ducking motion.</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Isabella Figueroa</td></tr>
</table>

<table>
<tr><td><b>Use Case ID:</b></td><td>UC-4</td></tr>
<tr><td><b>Use Case Name:</b></td><td>Encounter Obstacles.</td></tr>
<tr><td><b>Description:</b></td><td>Player collides with various obstacles, updating damage and score.</td></tr>
<tr><td><b>Actors:</b></td><td>Player.</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player crashes into an obstacle.</td></tr>
<tr><td><b>Post-conditions:</b></td><td>The player accumulates damage to the bike, and depending on object will gain or lose points from total score.</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Often.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player encounters frog.</td><td>If the player collided with the frog, then displays frog crashed. If the player jumped over the frog, then clear the frog sprite.</td></tr>
<tr><td><b>2</b></td><td>Player encounters bird.</td><td>If the player collided with the bird,</td></tr>
</table></td></tr>
</table>

		the system displays bird crashed. If the player ducked from the bird, then clear the bird sprite.
3	Player encounters a butterfly.	If the player collides with a butterfly, the system clears the butterfly sprite.
4		The system increments player's score.

Variations:	none
Exceptions:	none
Developer Notes:	Brandon Jacquez.

Use Case ID:	UC-5
Use Case Name:	Track Damage
Description:	Player has a full damage meter to start, when a user collides with an object, the damage meter will be updated.
Actors:	Player
Pre-conditions:	Player collides with an obstacle present on the course.
Post-conditions:	The player's damage meter reflects the damage accumulated by the recent collision.
Frequency of Use:	Often
Flow of Events:	

	Actor Action	**System Response**
1	Player collides with a "bad" object on the track.	System updates player's state to "collision".
2		System subtracts damage points, depending on the type of object hit.

<table>
<tr><td></td><td><table><tr><td><b>3</b></td><td></td><td>System displays updated damage on damage meter on screen.</td></tr></table></td></tr>
<tr><td><b>Variations:</b></td><td>Varying amounts of points taken off for:
Another Bike
Frog
Birds.</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Isabella Figueroa</td></tr>
</table>

<table>
<tr><td><b>Use Case ID:</b></td><td>UC-6</td></tr>
<tr><td><b>Use Case Name:</b></td><td>Track Time</td></tr>
<tr><td><b>Description:</b></td><td>A timer will be displayed on the screen, to indicate the time remaining in the race.</td></tr>
<tr><td><b>Actors:</b></td><td>Player</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player initiates game must be initiated.</td></tr>
<tr><td><b>Post-conditions:</b></td><td>Time will be displayed.</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Constantly throughout the game.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player initiates game.</td><td>System displays clock.</td></tr>
<tr><td><b>2</b></td><td></td><td>System updates clock to reflect time remaining in the game.</td></tr>
</table></td></tr>
<tr><td><b>Variations:</b></td><td>none</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
</table>

Developer Notes:	Dilara Madinger.

<table>
<tr><td><b>Use Case ID:</b></td><td>UC-7</td></tr>
<tr><td><b>Use Case Name:</b></td><td>Track Score.</td></tr>
<tr><td><b>Description:</b></td><td>Calculate score based on time elapsed and how much damage player’s bike accumulates.</td></tr>
<tr><td><b>Actors:</b></td><td>System</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player finished the race without bike breaking or timer going to 0</td></tr>
<tr><td><b>Post-conditions:</b></td><td>Player score is displayed to screen to be compared with high score board</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Once, at the end of each game</td></tr>
<tr><td><b>Flow of Events:</b></td><td>
<table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player finishes race.</td><td>System calculate score using damage and elapsed time.</td></tr>
<tr><td><b>2</b></td><td></td><td>System displays score.</td></tr>
<tr><td><b>3</b></td><td>User Inputs Name.</td><td>Add player’s new score to scores in database.</td></tr>
</table>
</td></tr>
<tr><td><b>Variations:</b></td><td>If the bike breaks the score should return 0</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Brandon Jacquez.</td></tr>
</table>

Use Case ID:	UC-8
Use Case Name:	Get Penalty.

<table>
<tr><td><b>Description:</b></td><td>If the damage meter fills then the game will end with a score of 0</td></tr>
<tr><td><b>Actors:</b></td><td>Player</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player collides with a “bad” object and takes damage, exceeding allowable amount.</td></tr>
<tr><td><b>Post-conditions:</b></td><td>Game ends with score of zero</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Uncommon</td></tr>
<tr><td><b>Flow of Events:</b></td><td>
<table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player damage meter exceeds limit.</td><td>System terminates current game.</td></tr>
<tr><td><b>2</b></td><td></td><td>System displays player score of zero.</td></tr>
</table>
</td></tr>
<tr><td><b>Variations:</b></td><td>none</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Isabella Figueroa</td></tr>
</table>

Use Case ID:	UC-9
Use Case Name:	View Scoreboard.
Description:	Player views the scoreboard with the highest scores and his/her score.
Actors:	Player
Pre-conditions:	Player chooses the option from the main menu to see the scoreboard. Player sees the scoreboard with his/her score at the end of the game.
Post-conditions:	Player presses the button on the screen to view the scoreboard. Player finished a game.

<table>
<tr><td><b>Frequency of Use:</b></td><td>Often.</td></tr>
<tr><td><b>Flow of Events:</b></td><td>
<table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player selects the option to see the scores.</td><td>Queries the database for the highest scores.</td></tr>
<tr><td><b>2</b></td><td></td><td>Display the scores in the score table.</td></tr>
<tr><td><b>3</b></td><td>The game session finishes.</td><td>Queries the database for the highest scores.</td></tr>
<tr><td><b>4</b></td><td></td><td>System displays the highest scores and the last session score in the sorted order in the score table.</td></tr>
</table>
</td></tr>
<tr><td><b>Variations:</b></td><td>none.</td></tr>
<tr><td><b>Exceptions:</b></td><td>none.</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Dilara Madinger.</td></tr>
</table>

Use Case ID:	UC-10
Use Case Name:	Have Audio.
Description:	Have some music playing while in menus or racing as well as sounds for jumping/landing, hitting a frog, hitting a duck, etc.
Actors:	System
Pre-conditions:	Player initializes game.
Post-conditions:	Sound is output given the state of the game (menu, racing, collision detected)
Frequency of	Ubiquitous

<table>
<tr><td><b>Use:</b></td><td></td></tr>
<tr><td><b>Flow of Events:</b></td><td><table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Hit frog</td><td>Frog croaking noise</td></tr>
<tr><td><b>2</b></td><td>Hit duck</td><td>Duck quacking noise</td></tr>
<tr><td><b>3</b></td><td>Enter menu</td><td>Play calm music</td></tr>
<tr><td><b>4</b></td><td>Enter race</td><td>Play racing music</td></tr>
</table></td></tr>
<tr><td><b>Variations:</b></td><td>none</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Brandon Jacquez</td></tr>
</table>

<table>
<tr><td><b>Use Case ID:</b></td><td>UC-11</td></tr>
<tr><td><b>Use Case Name:</b></td><td>Encounter Another Bike.</td></tr>
<tr><td><b>Description:</b></td><td>Player encounters another bike that is going in the opposite direction. The bike will randomly choose to jump or keep going straight. Player must avoid the crash with the bike by either jumping or going straight.</td></tr>
<tr><td><b>Actors:</b></td><td>Player.</td></tr>
<tr><td><b>Pre-conditions:</b></td><td>Player has collected some butterflies and avoided some frogs or birds.</td></tr>
<tr><td><b>Post-conditions:</b></td><td>In the event of a crash player loses some points, gains damage. In the event of no crash, player continues the track.</td></tr>
<tr><td><b>Frequency of Use:</b></td><td>Common.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><table>
<tr><th></th><th>Actor Action</th><th>System Response</th></tr>
<tr><td><b>1</b></td><td>Player avoids the crash.</td><td>System removes bike’s sprite,</td></tr>
</table></td></tr>
</table>

		when it approaches the left side of the screen.
2	Player crashes into the bike.	System outputs a crash sound.
3		System decrements the score.
4		System increments damage.
		System displays two crashed bikes.
		System redraws the bikes at the beginning of their encounter.

Variations:	none
Exceptions:	none
Developer Notes:	Dilara Madinger

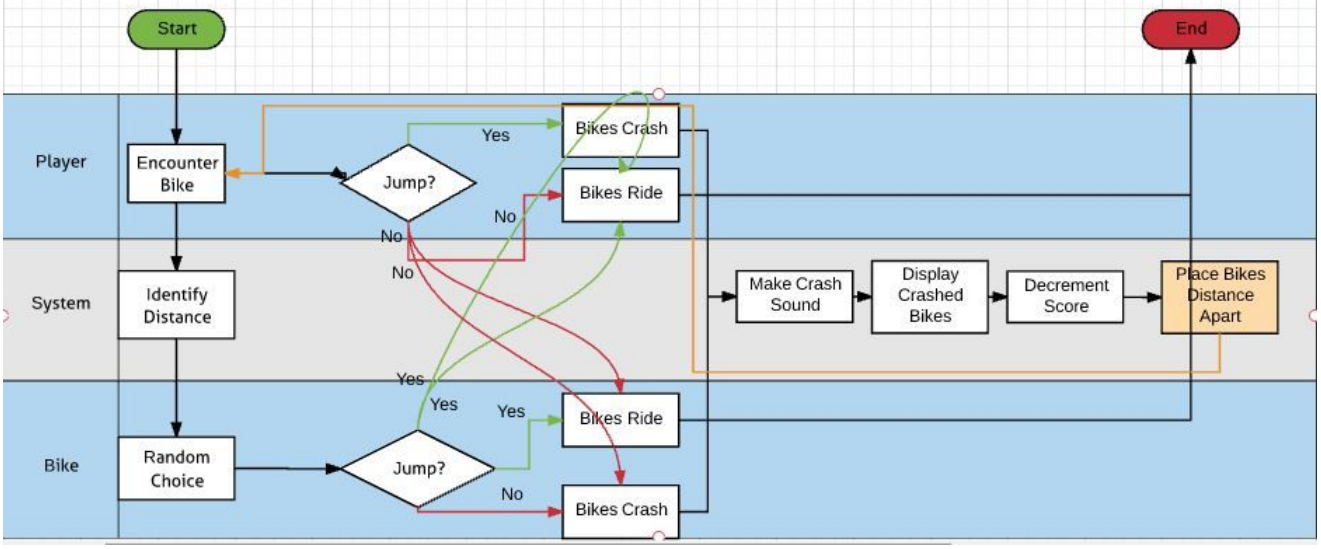
Use Case ID:	UC-12
Use Case Name:	Input Nickname
Description:	User inputs his/her preferred nickname at the beginning of the game.
Actors:	Player
Pre-conditions:	User initiates the game.
Post-conditions:	The score is saved under the nickname.
Frequency of Use:	Common.
Flow of Events:	

	Actor Action	**System Response**
1	User presses the start button.	Prints out a window for the

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td>nickname.</td></tr>
<tr><td><b>2</b></td><td>User types in a nickname and presses OK.</td><td>System saves the nickname and associates the current game with it.</td></tr>
</table>
</td></tr>
<tr><td><b>Variations:</b></td><td>none</td></tr>
<tr><td><b>Exceptions:</b></td><td>none</td></tr>
<tr><td><b>Developer Notes:</b></td><td>Dilara Madinger</td></tr>
</table>

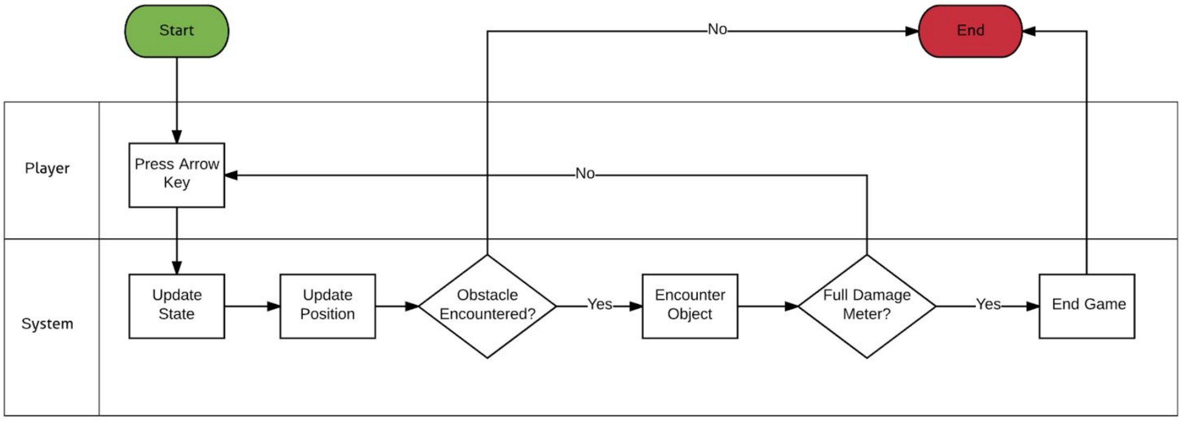
Activity Diagrams

Requirement ID#: UR-11
Use Case ID #: UC-11
Use Case name: Encounter Another Bike
Name: Dilara Madinger



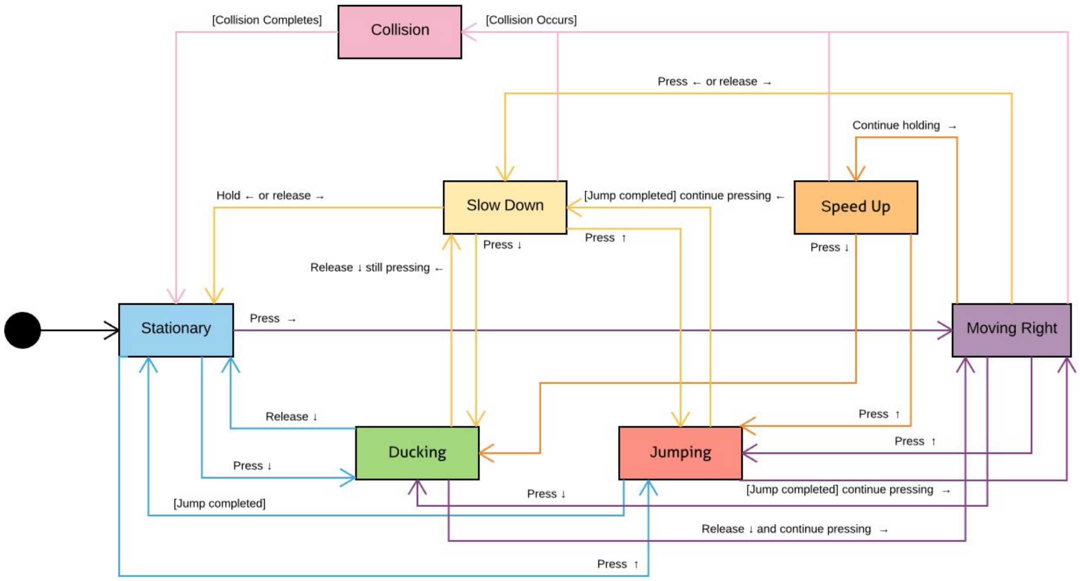
Requirement ID#: UR - 3
Use Case ID #: UC - 3
Use Case name: Move Player
Name: Isabella Figueroa

Trivial Use Case, See Move Player State Diagram for addiational explanation

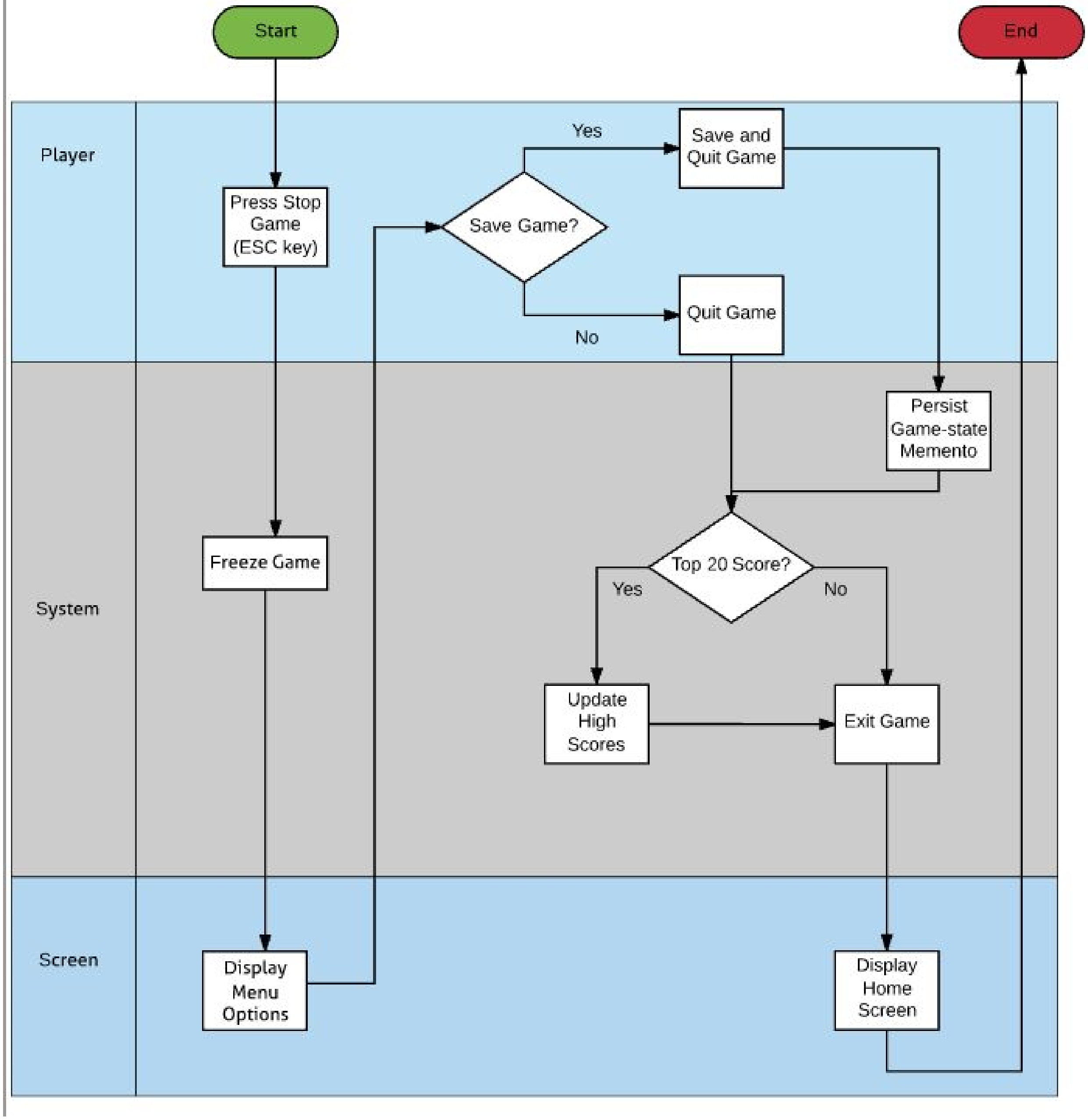


State Diagram

Requirement ID#: UR - 3
Use Case ID #: UC - 3
Use Case name: Move Player
Name: Isabella Figueroa

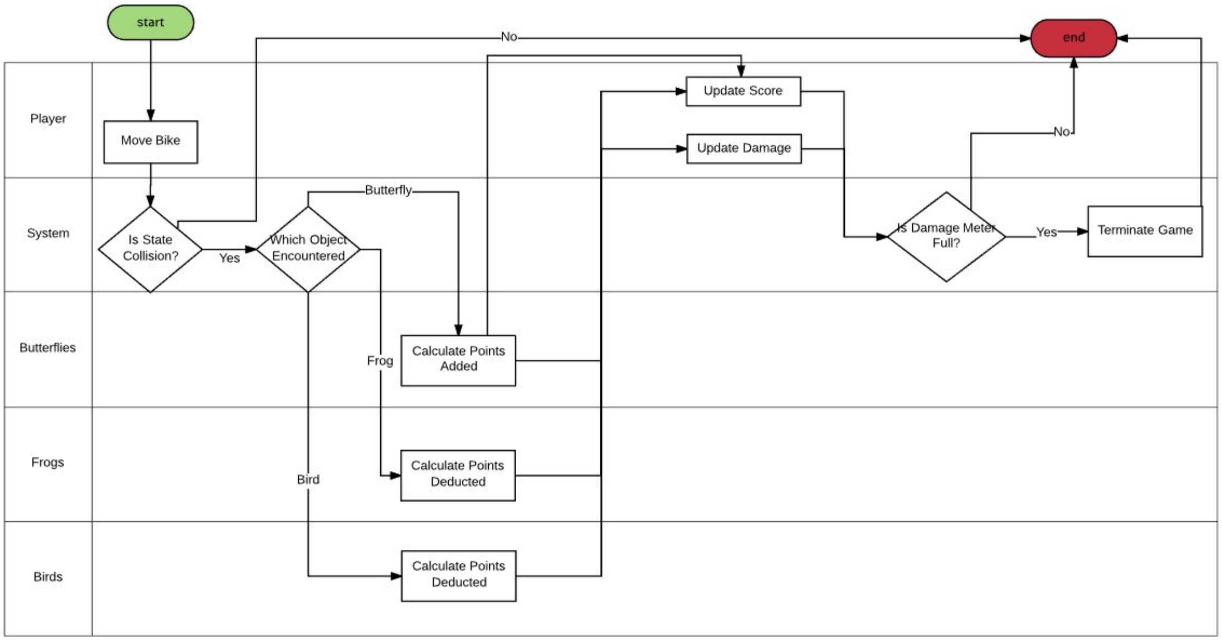


Requirement ID#: UR-1
Use Case ID #: UC-1
Use Case name: Stop Game
Name: Brandon Barrett



Requirement ID # UR 4
Use Case ID # UC 4
Use Case Name: Encounter Obstacles
Name: Brandon Jacquez

Encounter Obstacles - Brandon Jacquez



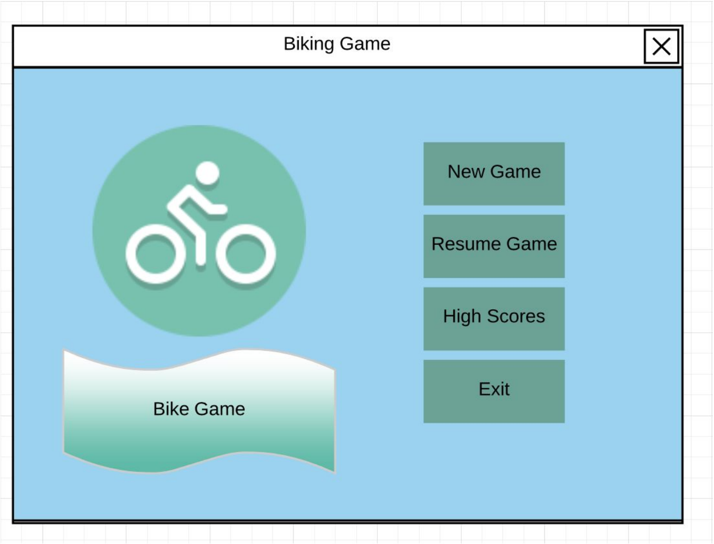
Data Storage

For storage we will use a file that will have two fields for the nickname of the player and his/her score. The fields will be separated by a comma. The user will be able to obtain nicknames and their scores by clicking View Scores on the Menu page.

UI Mockups

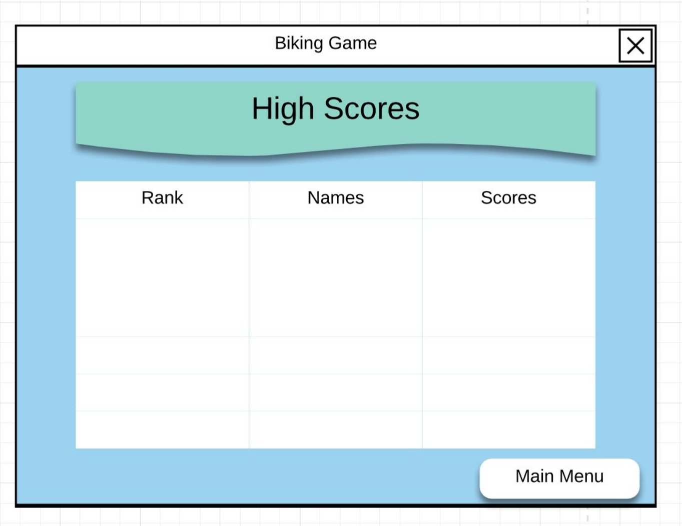
Opening Screen

Game opens on the following screen.



High Scores : Data Display

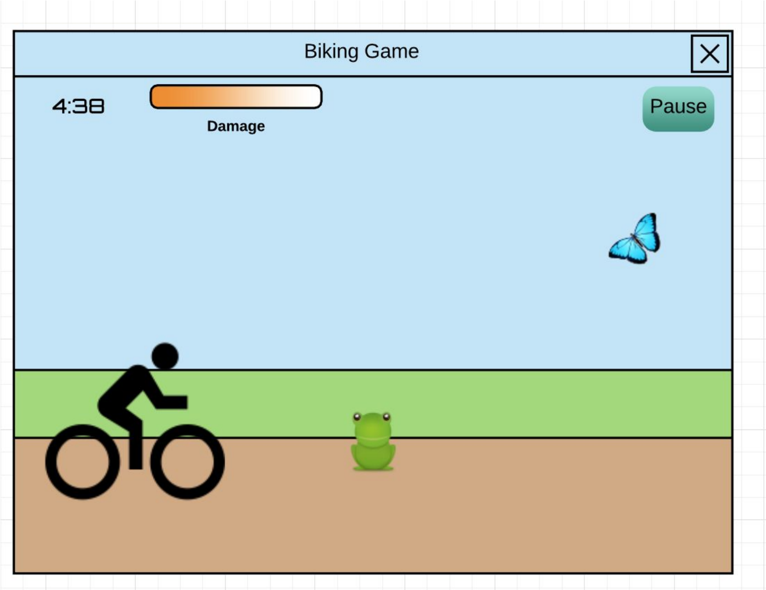
If the player selects *High Scores* the following screen displays, populated with names and scores saved to the database.



Rank	Names	Scores

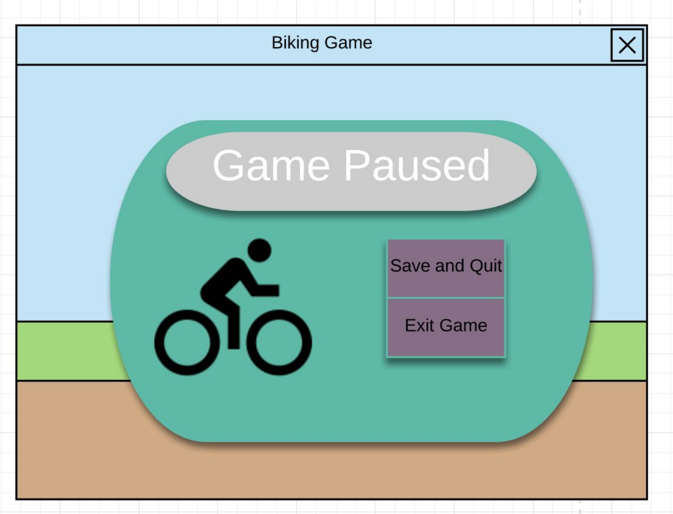
Game Play Screen

When the player selects *New Game* or *Resume* game a screen similar to the following will display.



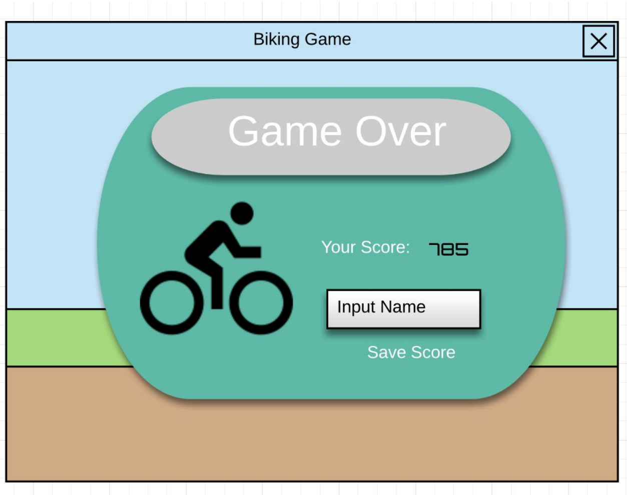
Pause Screen

If the player clicks the *Pause* button during the game the following will display with options to quit.

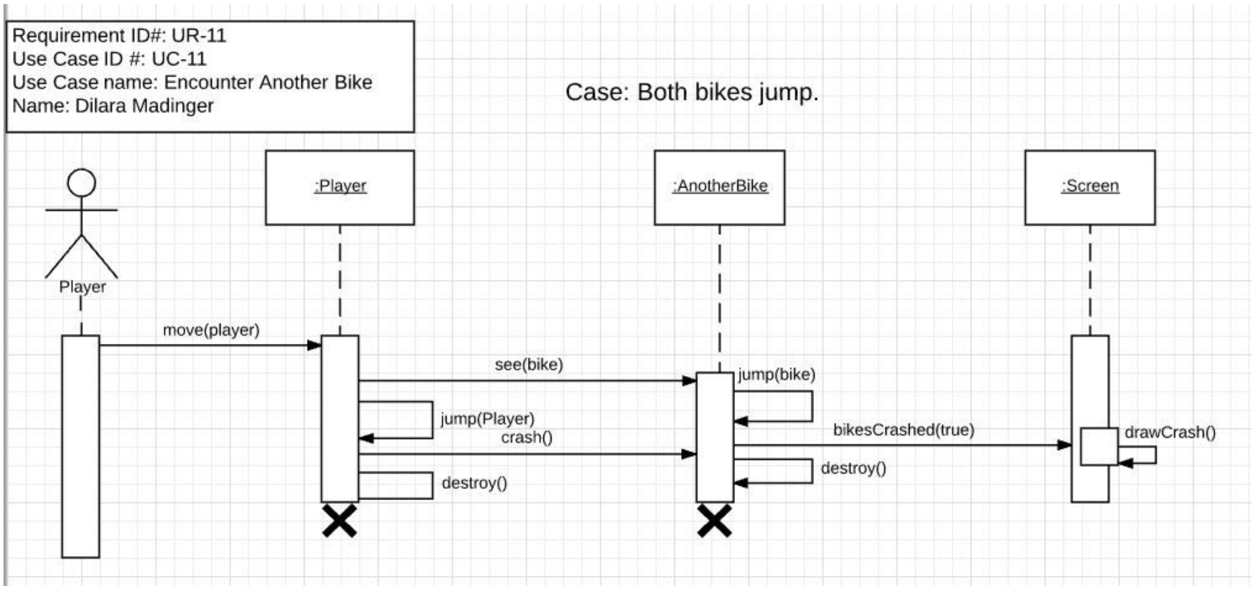


Game Over Screen

If at any time the game ends, the following will display, prompting the player to input a name and submit their score to be saved under High Scores.

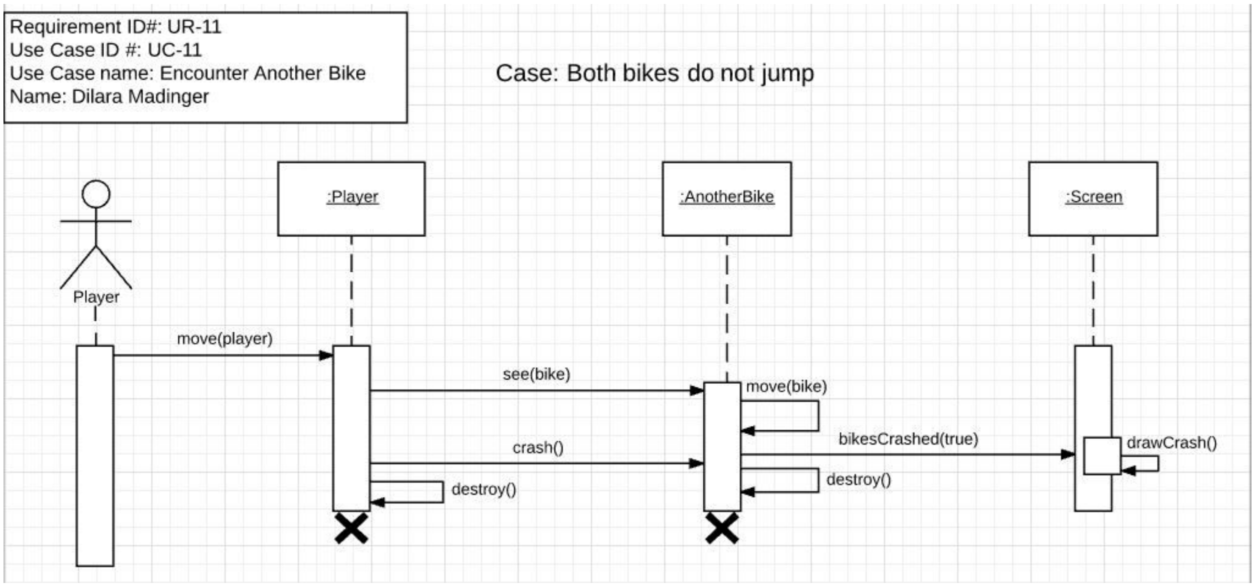


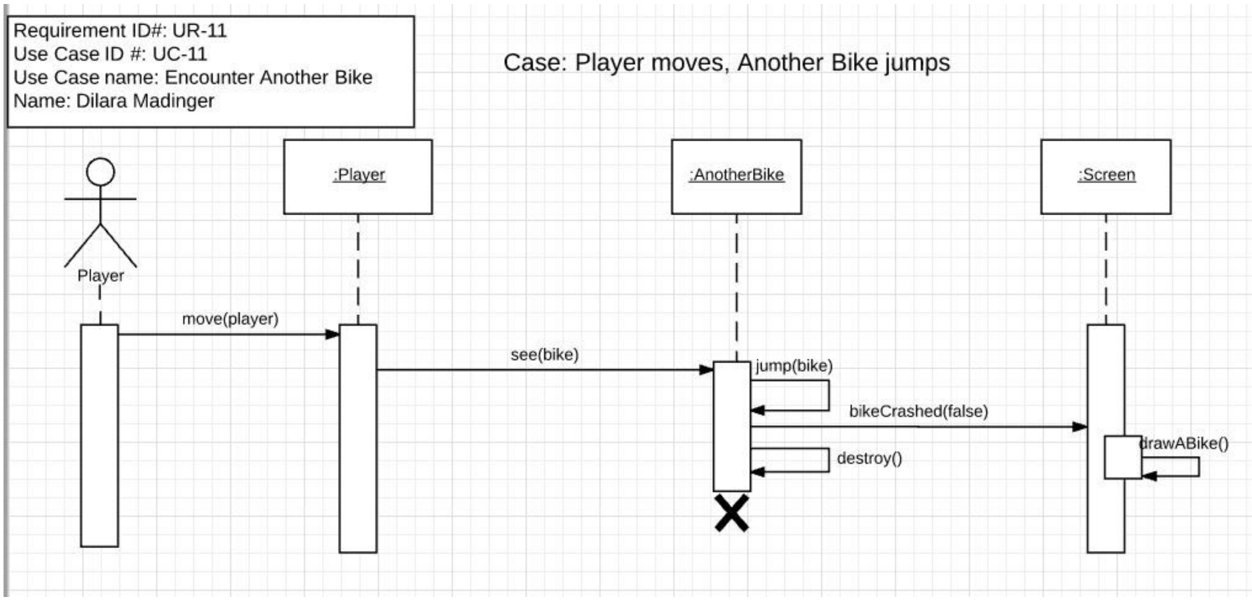
User Interactions

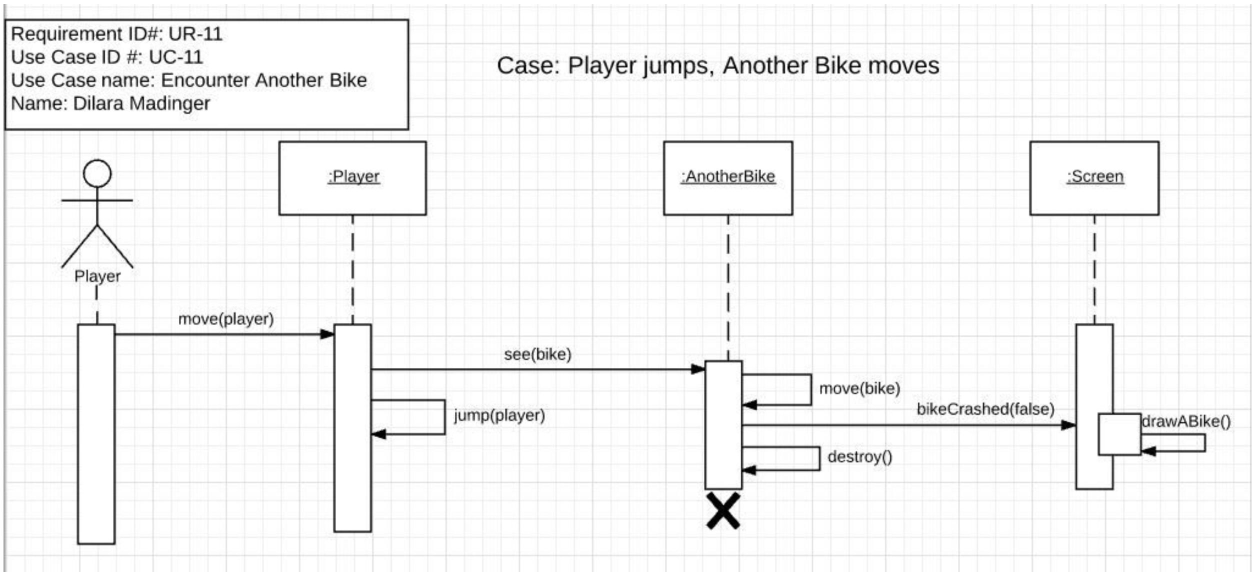


Requirement ID#: UR-11
Use Case ID #: UC-11
Use Case name: Encounter Another Bike
Name: Dilara Madinger

Case: Both bikes jump.


Requirement ID#: UR-11
Use Case ID #: UC-11
Use Case name: Encounter Another Bike
Name: Dilara Madinger
Case: Both bikes do not jump
:Player
:AnotherBike
:Screen
Player
move(player)
see(bike)
move(bike)
bikesCrashed(true)
drawCrash()
crash()
destroy()
destroy()


Requirement ID#: UR-11
Use Case ID #: UC-11
Use Case name: Encounter Another Bike
Name: Dilara Madinger
Case: Player moves, Another Bike jumps
:Player
:AnotherBike
:Screen
Player
move(player)
see(bike)
jump(bike)
bikeCrashed(false)
drawABike()
destroy()

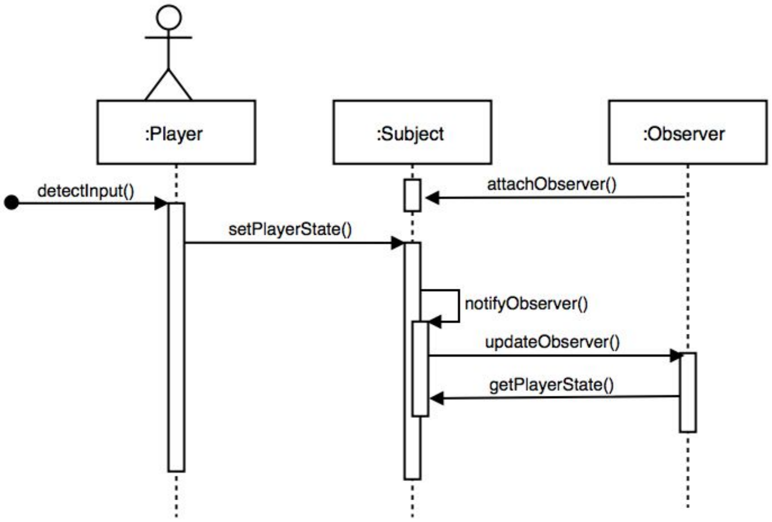


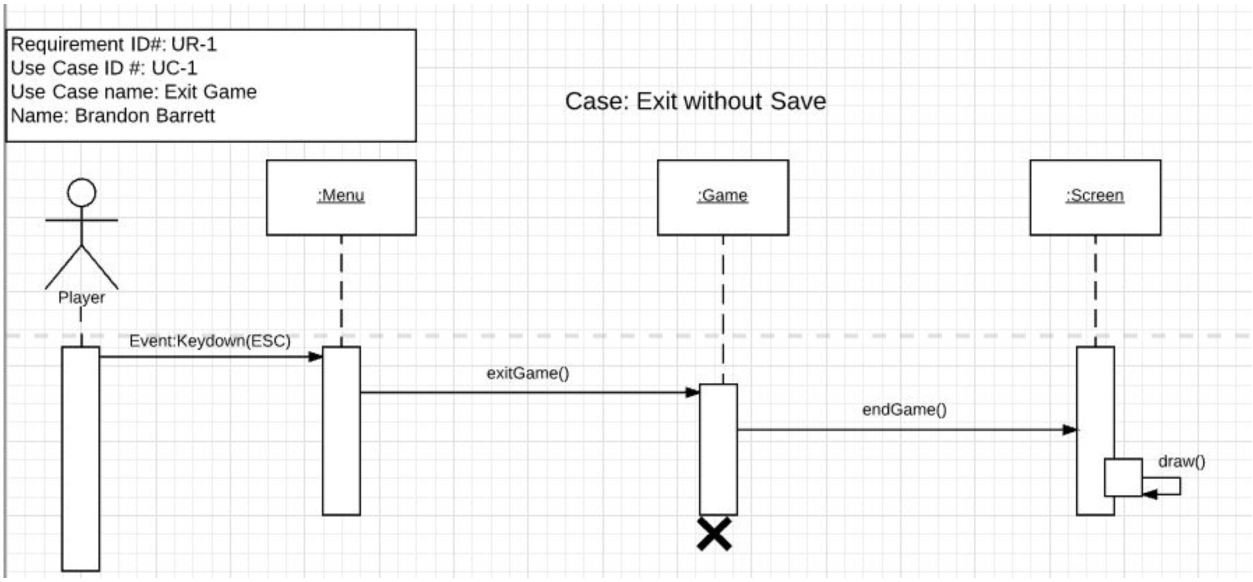
Requirement ID#: UR-11
Use Case ID #: UC-11
Use Case name: Encounter Another Bike
Name: Dilara Madinger

Case: Player jumps, Another Bike moves

Requirement ID#: UR - 3
Use Case ID#: UC-3
Use Case Name: Move Player
Name: Isabella Figueroa

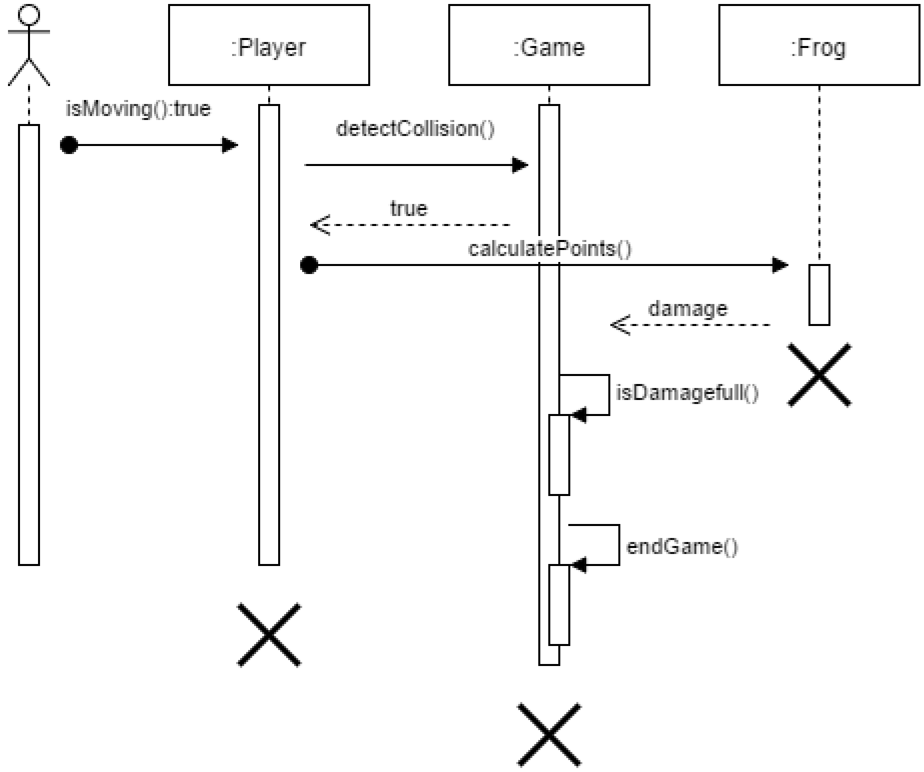
Please note that this version of Move Player is depicted using the Observer Design Pattern



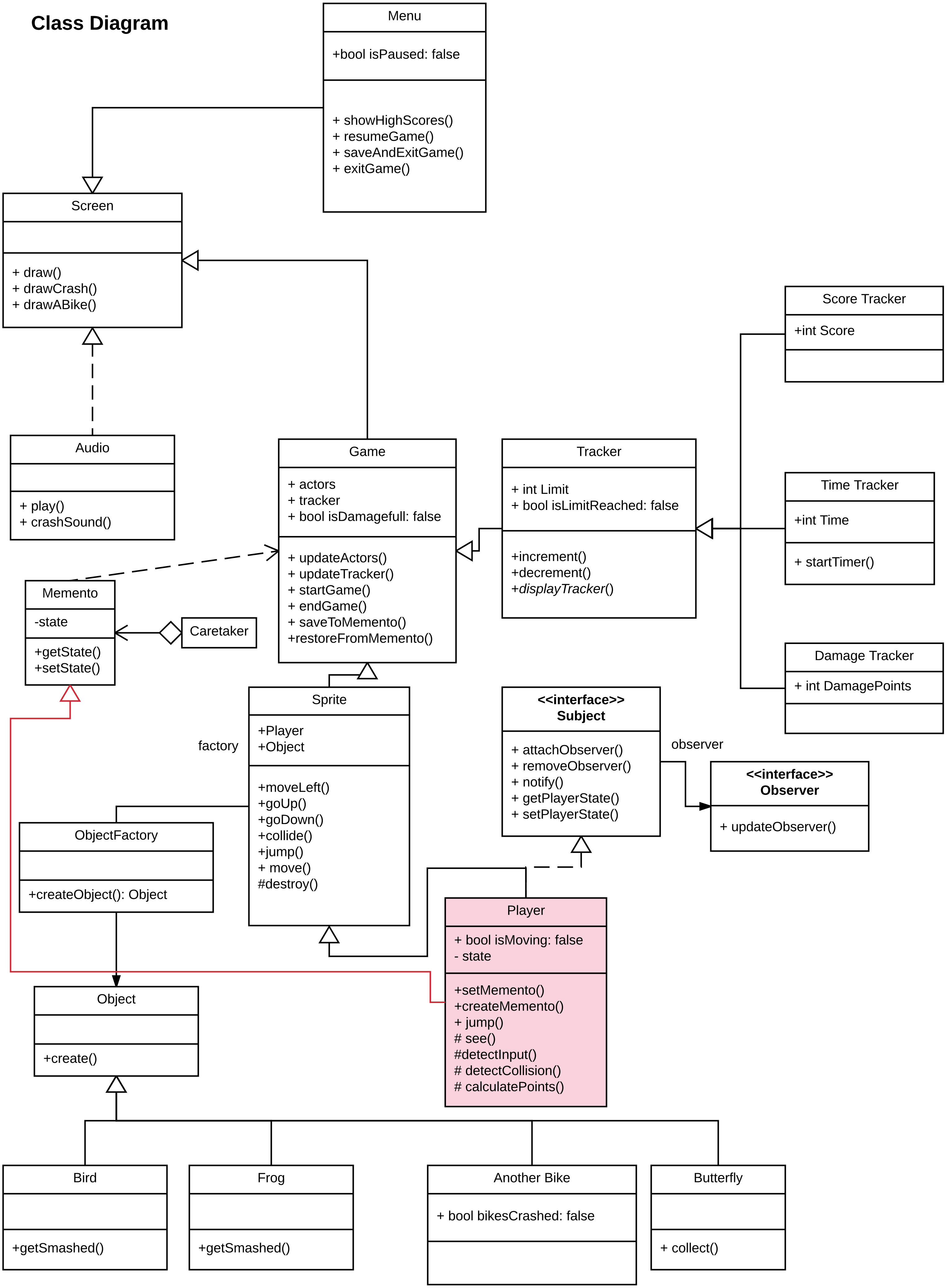


Requirement ID#: UR-1
Use Case ID #: UC-1
Use Case name: Exit Game
Name: Brandon Barrett

Requirement ID #- UR 7
Use Case ID # - UC 7
Use Case Name: Encounter Obstacles
Name: Brandon Jacquez



Class Diagram


Menu
+bool isPaused: false
+ showHighScores()
+ resumeGame()
+ saveAndExitGame()
+ exitGame()
Screen
+ draw()
+ drawCrash()
+ drawABike()
Score Tracker
+int Score
Audio
+ play()
+ crashSound()
Game
+ actors
+ tracker
+ bool isDamagefull: false
+ updateActors()
+ updateTracker()
+ startGame()
+ endGame()
+ saveToMemento()
+restoreFromMemento()
Tracker
+ int Limit
+ bool isLimitReached: false
+increment()
+decrement()
+displayTracker()
Time Tracker
+int Time
+ startTimer()
Memento
-state
+getState()
+setState()
Caretaker
Damage Tracker
+ int DamagePoints
Sprite
+Player
+Object
+moveLeft()
+goUp()
+goDown()
+collide()
+jump()
+ move()
#destroy()
factory
<<interface>>
Subject
+ attachObserver()
+ removeObserver()
+ notify()
+ getPlayerState()
+ setPlayerState()
observer
<<interface>>
Observer
+ updateObserver()
ObjectFactory
+createObject(): Object
Player
+ bool isMoving: false
- state
+setMemento()
+createMemento()
+ jump()
# see()
#detectInput()
# detectCollision()
# calculatePoints()
Object
+create()
Bird
+getSmashed()
Frog
+getSmashed()
Another Bike
+ bool bikesCrashed: false
Butterfly
+ collect()