CMPE 202 Sprint #2: Team Project

Report Submitted by:

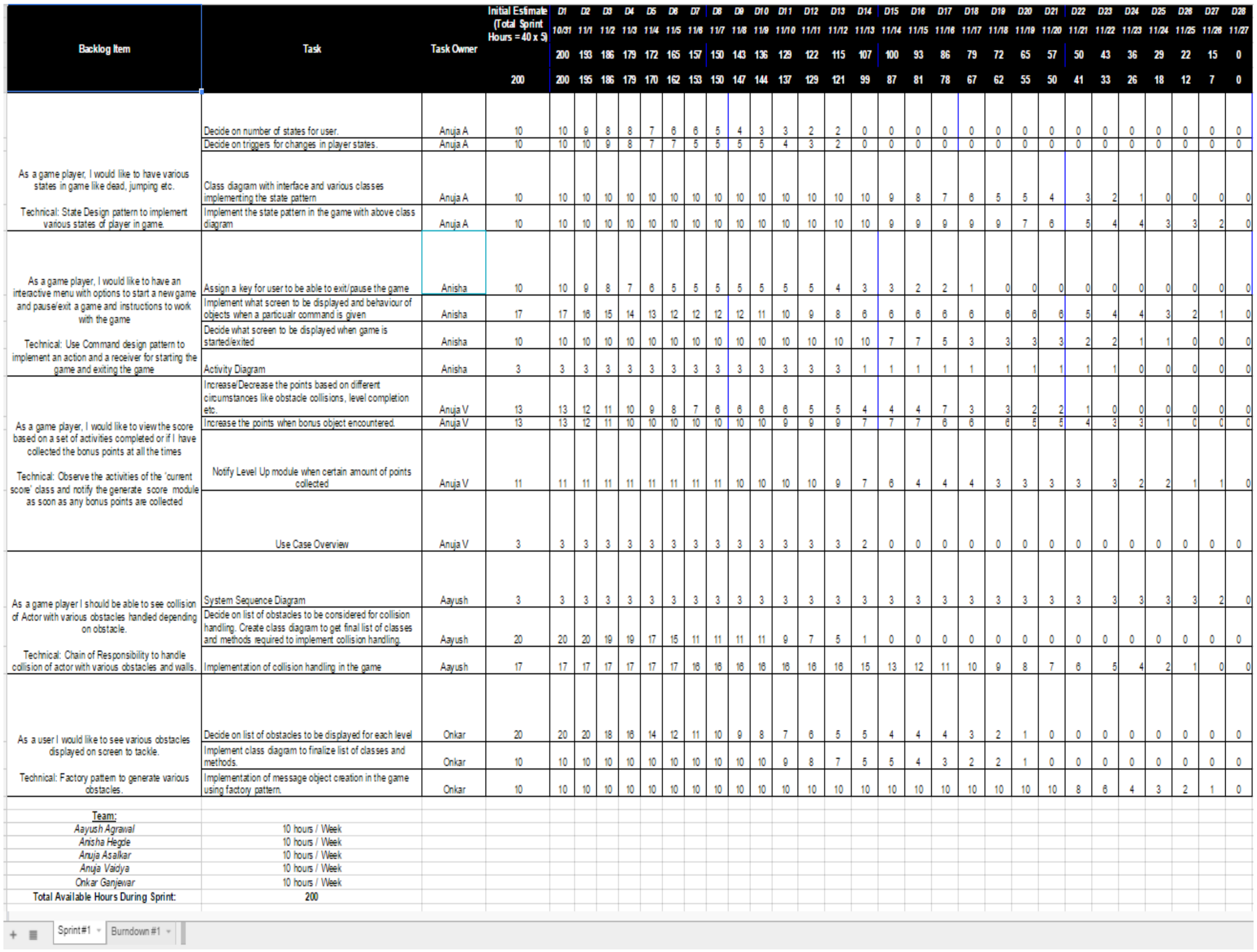
Anuja Asalkar (010734841)
Anuja Vaidya (009982674)
Anisha Hegde (010726430)
Aayush Agrawal (010691382)
Onkar Ganjewar (010675782)

Department of Software Engineering

Submitted To:

Prof. Paul Nguyen

User Stories, Task Breakdown, Team member assignments and initial estimates:

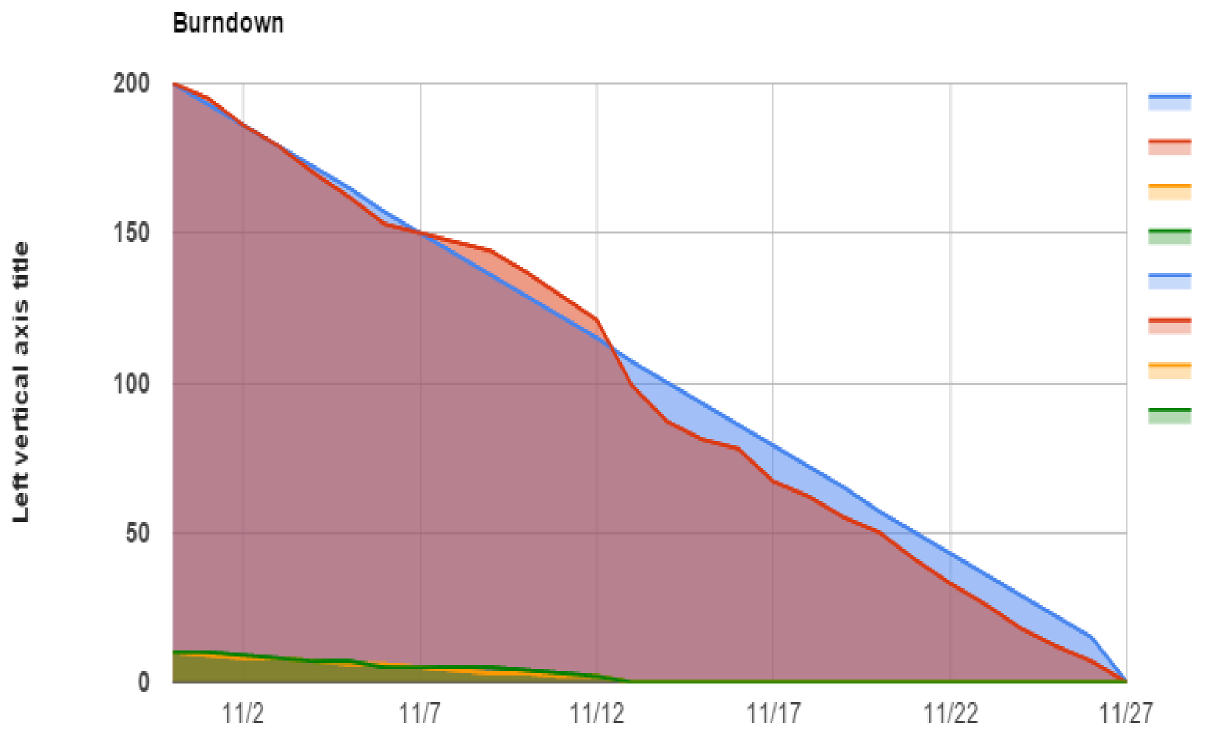


Backlog Item	Task	Task Owner	Initial Estimate (Total Sprint Hours = 40 x 5)	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28
				10/31	11/1	11/2	11/3	11/4	11/5	11/6	11/7	11/8	11/9	11/10	11/11	11/12	11/13	11/14	11/15	11/16	11/17	11/18	11/19	11/20	11/21	11/22	11/23	11/24	11/25	11/26	11/27
				200	193	186	179	172	165	157	150	143	136	129	122	115	107	100	93	86	79	72	65	57	50	43	36	29	22	15	0
			200	200	195	186	179	170	162	153	150	147	144	137	129	121	99	87	81	78	67	62	55	50	41	33	26	18	12	7	0
As a game player, I would like to have various states in game like dead, jumping etc. Technical: State Design pattern to implement various states of player in game.	Decide on number of states for user.	Anuja A	10	10	9	8	8	7	6	6	5	4	3	3	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Decide on triggers for changes in player states.	Anuja A	10	10	10	9	8	7	7	5	5	5	5	4	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Class diagram with interface and various classes implementing the state pattern	Anuja A	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	9	8	7	6	5	5	4	3	2	1	0	0	0	0
	Implement the state pattern in the game with above class diagram	Anuja A	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	9	9	9	9	9	7	6	5	4	4	3	3	2	0
As a game player, I would like to have an interactive menu with options to start a new game and pause/exit a game and instructions to work with the game Technical: Use Command design pattern to implement an action and a receiver for starting the game and exiting the game	Assign a key for user to be able to exit/pause the game	Anisha	10	10	9	8	7	6	5	5	5	5	5	5	5	4	3	3	2	2	1	0	0	0	0	0	0	0	0	0	0
	Implement what screen to be displayed and behaviour of objects when a particualr command is given	Anisha	17	17	16	15	14	13	12	12	12	12	11	10	9	8	6	6	6	6	6	6	6	6	5	4	4	3	2	1	0
	Decide what screen to be displayed when game is started/exited	Anisha	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	7	7	5	3	3	3	3	2	2	1	1	0	0	0
	Activity Diagram	Anisha	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
As a game player, I would like to view the score based on a set of activities completed or if I have collected the bonus points at all the times Technical: Observe the activities of the 'current score' class and notify the generate score module as soon as any bonus points are collected	Increase/Decrease the points based on different circumstances like obstacle collisions, level completion etc.	Anuja V	13	13	12	11	10	9	8	7	6	6	6	6	5	5	4	4	4	7	3	3	2	2	1	0	0	0	0	0	0
	Increase the points when bonus object encountered.	Anuja V	13	13	12	11	10	10	10	10	10	10	10	9	9	9	7	7	7	6	6	6	5	5	4	3	3	1	0	0	0
	Notify Level Up module when certain amount of points collected	Anuja V	11	11	11	11	11	11	11	11	11	10	10	10	10	9	7	6	4	4	4	3	3	3	3	3	2	2	1	1	0
	Use Case Overview	Anuja V	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
As a game player I should be able to see collision of Actor with various obstacles handled depending on obstacle. Technical: Chain of Responsibility to handle collision of actor with various obstacles and walls.	System Sequence Diagram	Aayush	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	0
	Decide on list of obstacles to be considered for collision handling. Create class diagram to get final list of classes and methods required to implement collision handling.	Aayush	20	20	20	19	19	17	15	11	11	11	11	9	7	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Implementation of collision handling in the game	Aayush	17	17	17	17	17	17	17	16	16	16	16	16	16	16	15	13	12	11	10	9	8	7	6	5	4	2	1	0	0
As a user I would like to see various obstacles displayed on screen to tackle. Technical: Factory pattern to generate various obstacles.	Decide on list of obstacles to be displayed for each level	Onkar	20	20	20	18	16	14	12	11	10	9	8	7	6	5	5	4	4	4	3	2	1	0	0	0	0	0	0	0	0
	Implement class diagram to finalize list of classes and methods.	Onkar	10	10	10	10	10	10	10	10	10	10	10	9	8	7	5	5	4	3	2	2	1	0	0	0	0	0	0	0	0
	Implementation of message object creation in the game using factory pattern.	Onkar	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	8	6	4	3	2	1	0

Team:	
Aayush Agrawal	10 hours / Week
Anisha Hegde	10 hours / Week
Anuja Asalkar	10 hours / Week
Anuja Vaidya	10 hours / Week
Onkar Ganjewar	10 hours / Week
Total Available Hours During Sprint:	**200**

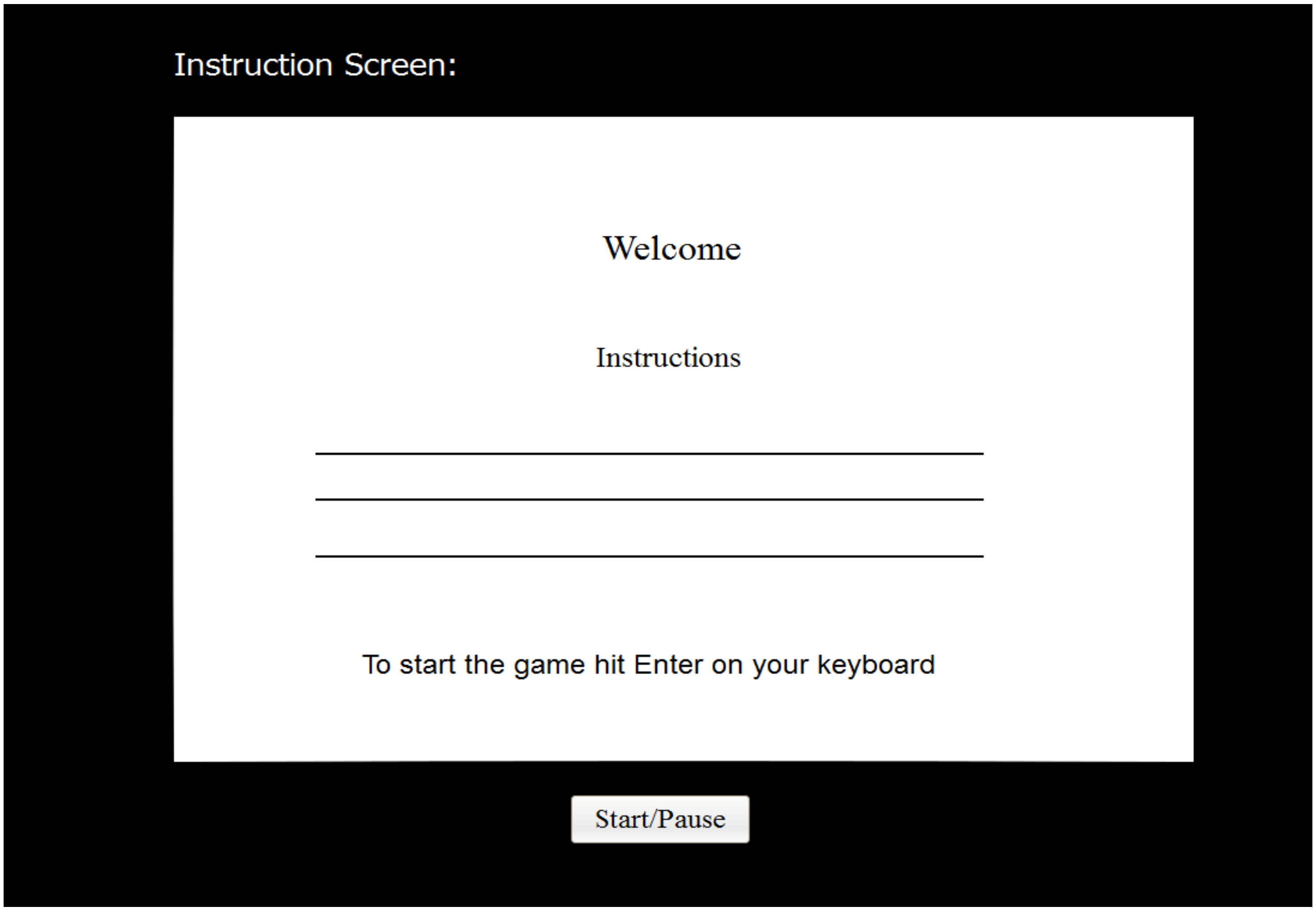
Sprint#1 Burndown #1

Burn Down Chart:

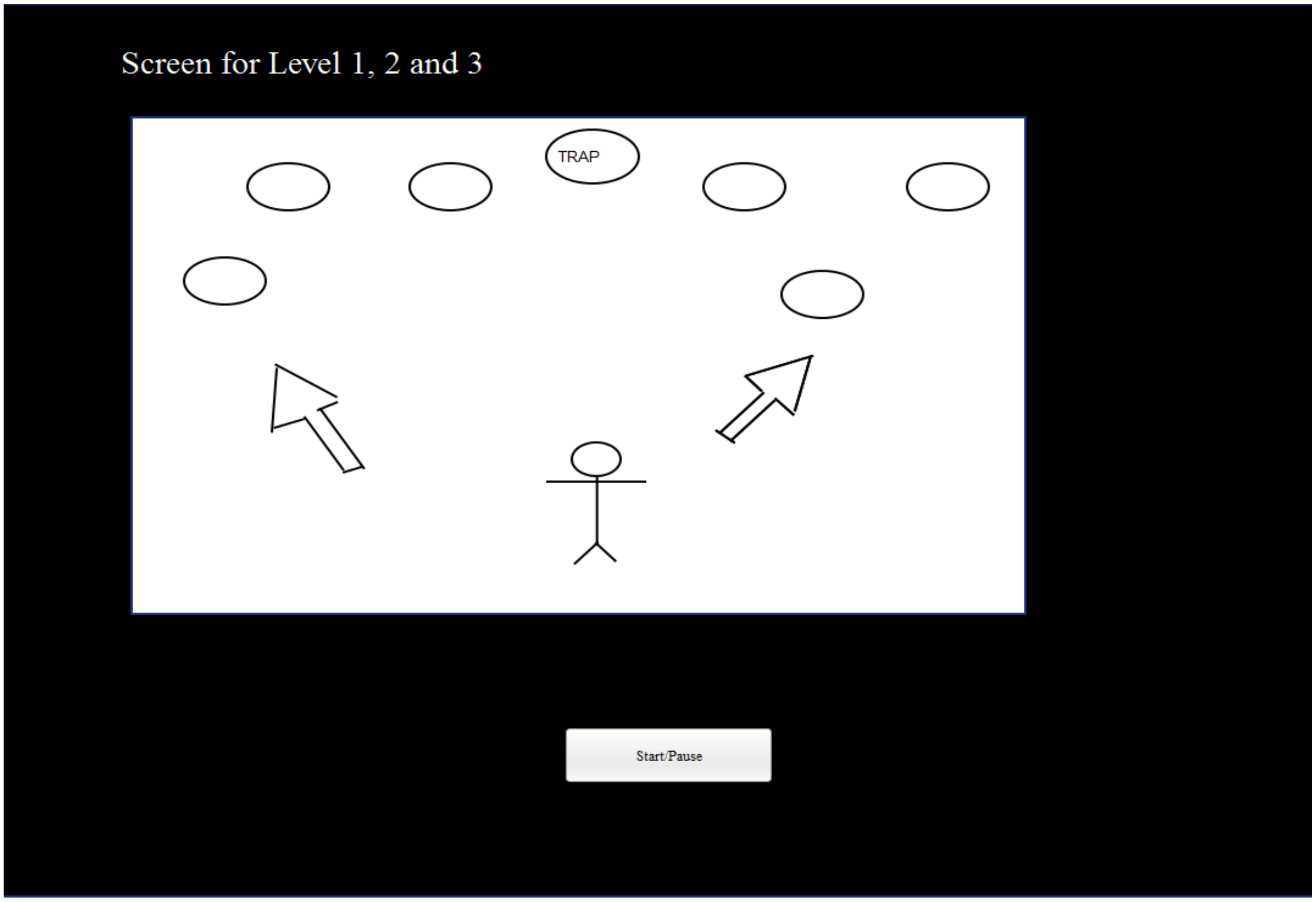


UI wireframes:

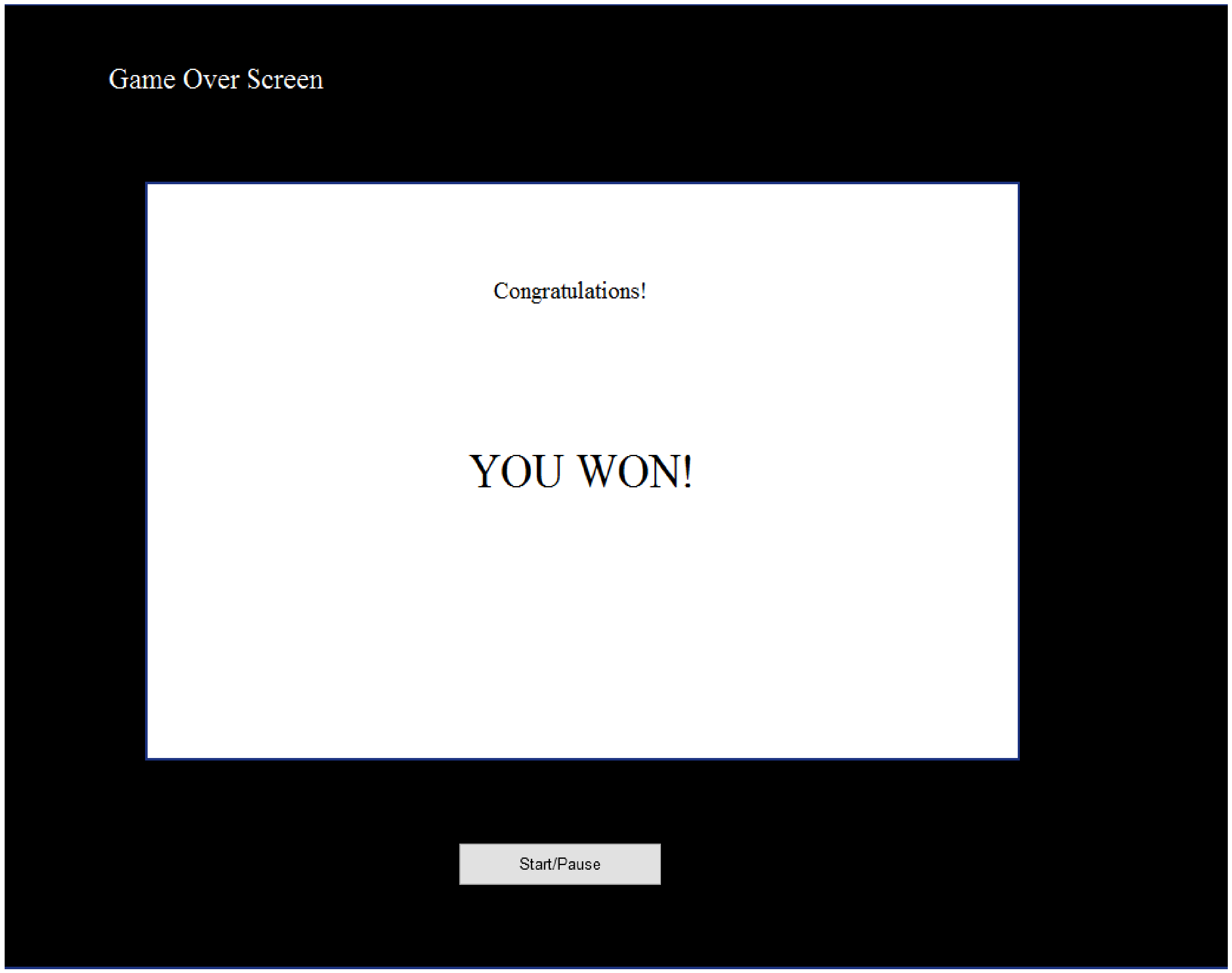
- Instruction Page



- Game Screen

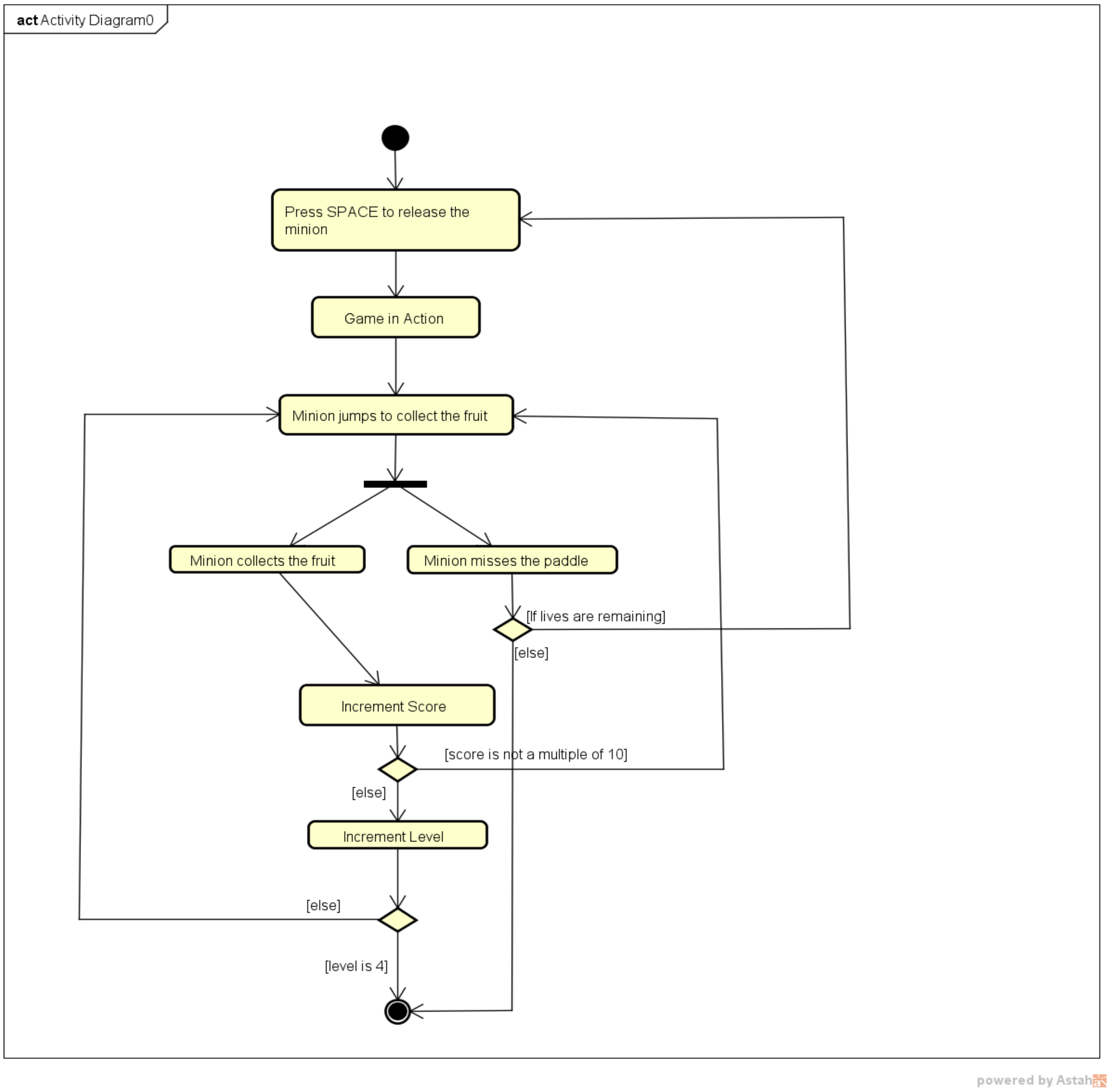


- Game over Page

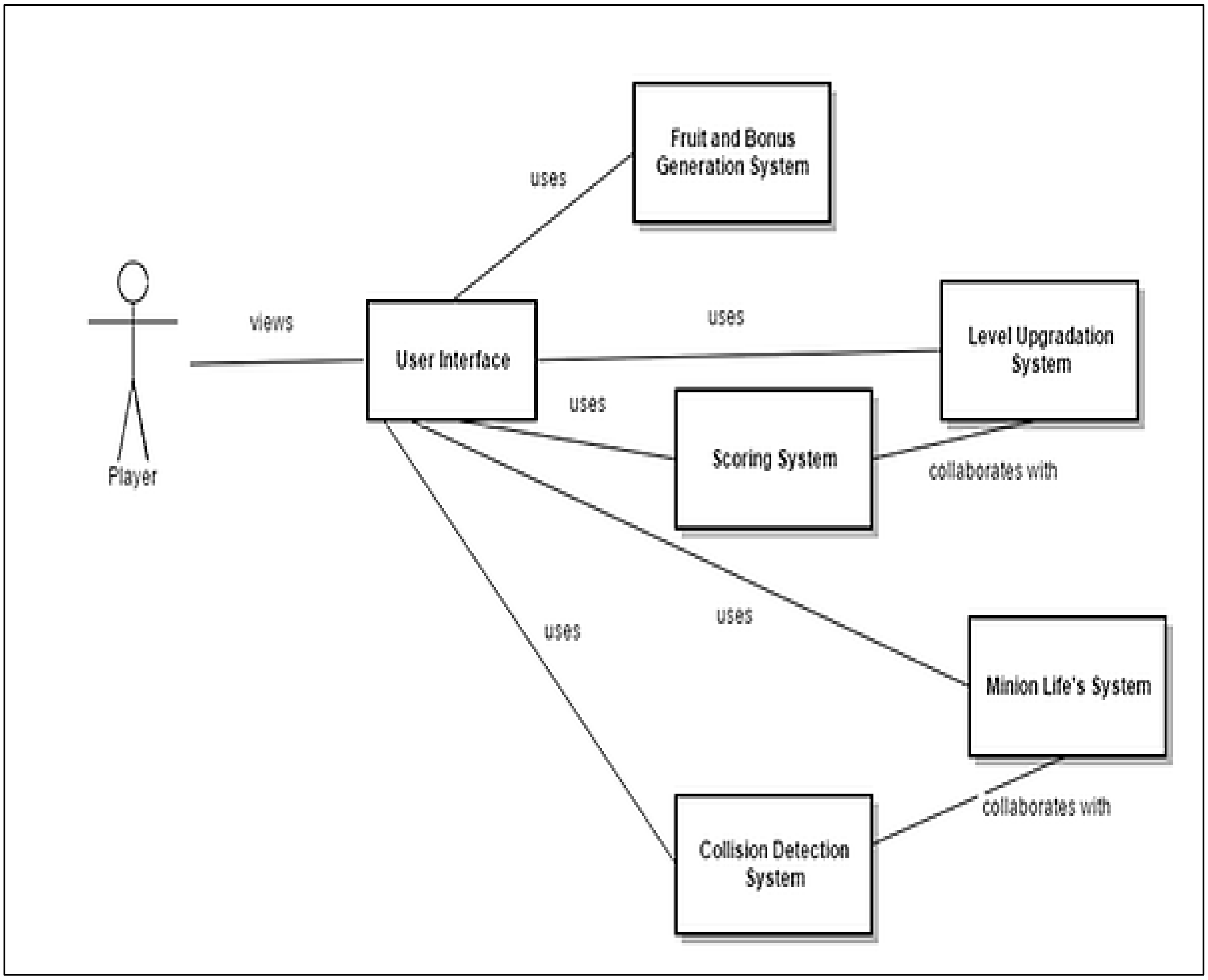


UML Analysis Model

- Activity Diagram



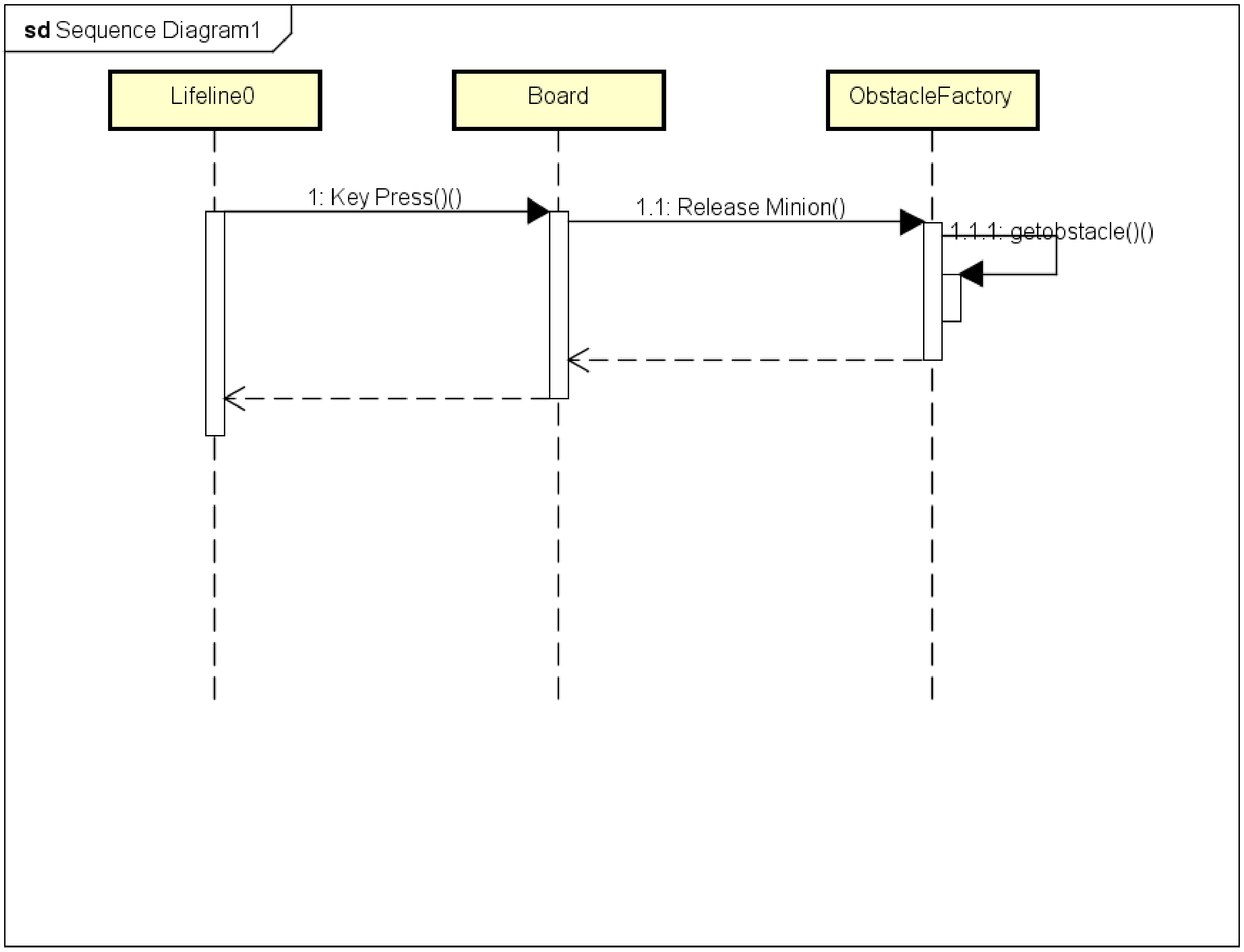
- Use Case Overview Diagram



Use Case Specification

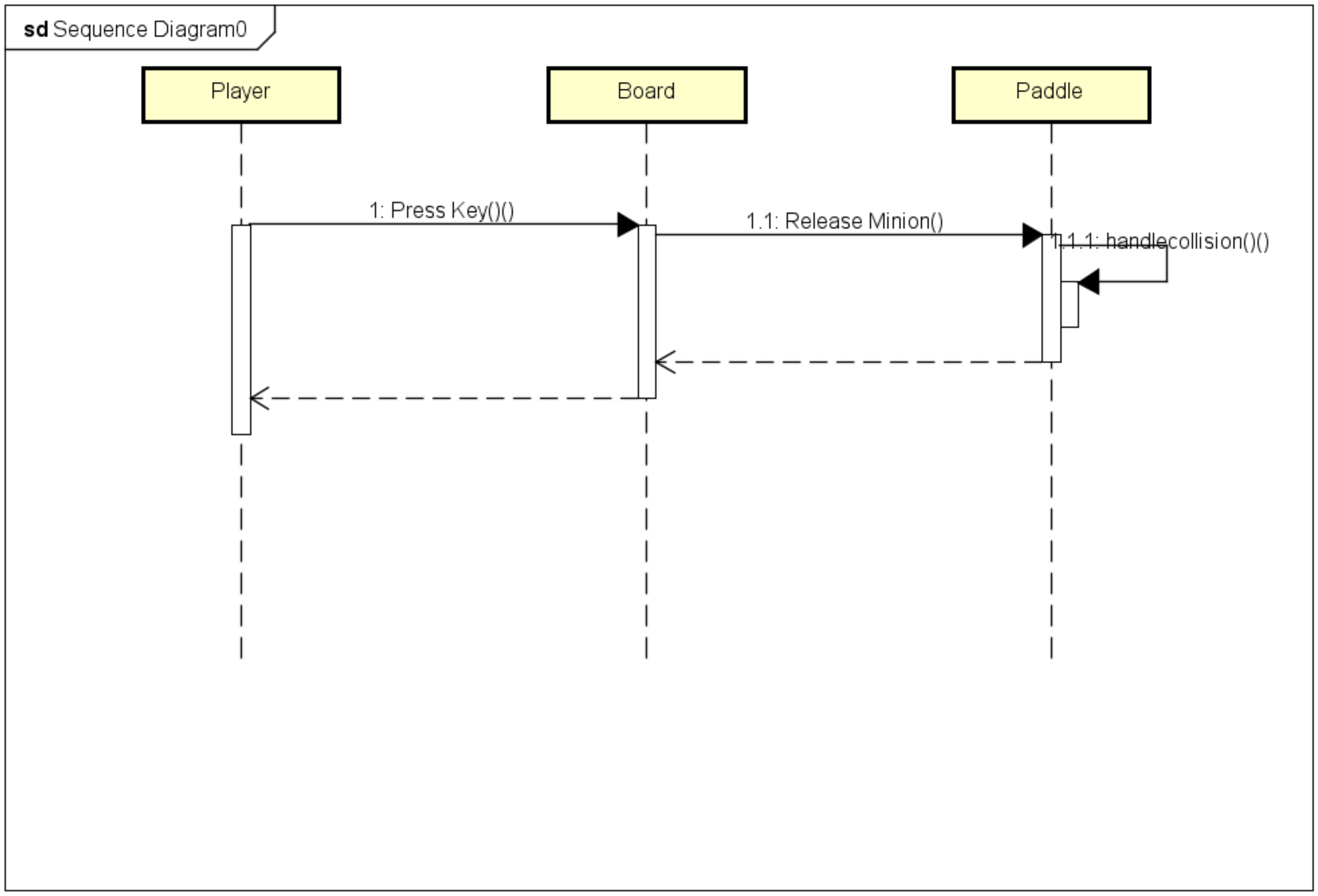
- Collect Fruits

Use Case Name	**Collect Fruits**
Description	The Minion should collect all the fruits in existing level to move to next level.
Actor	Player
Precondition	The minion should be alive and player should be able to move the paddle to left and right side.
Trigger	Space key is pressed to release the minion from paddle and start jumping
Basic Flow	To collect fruit Minion should collide with fruit object displayed on game screen.
Failure Condition	Minion not able to collect fruit even after collision and hence not able to move to next level.
Successful End Condition	All the fruits are collected and Player moves to next level


sd Sequence Diagram1
Lifeline0
Board
ObstacleFactory
1: Key Press()()
1.1: Release Minion()
1.1.1: getobstacle()()

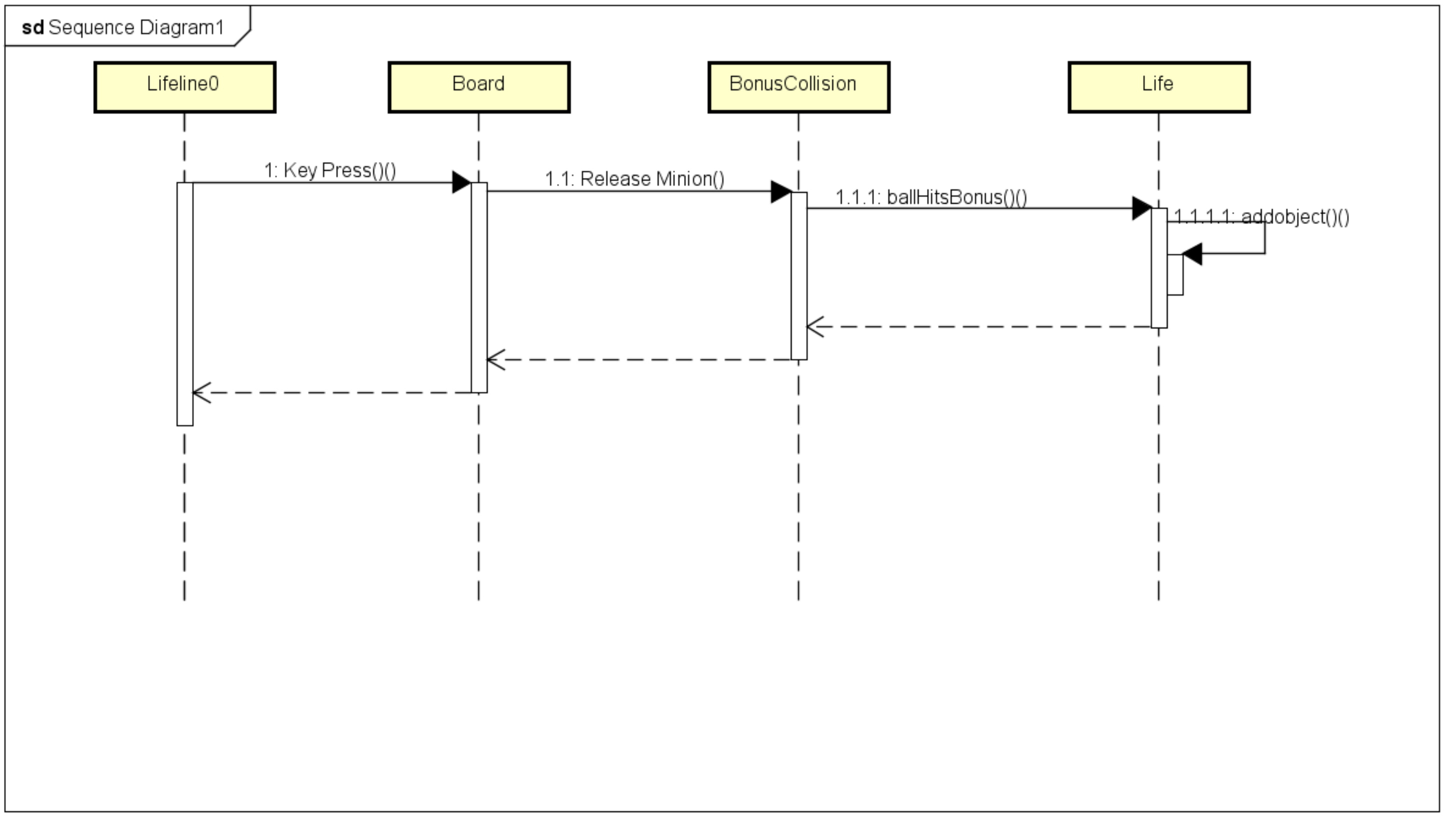
- Game Controls

Use Case Name	Game Controls
Description	The Minion should be able to jump from the paddle and move all around the game and the player should be able to move the paddle
Actor	Player
Precondition	The player should start the game, with minion on the paddle.
Trigger	The player presses Left or Right key to move the paddle and prevents minion from falling down.
Basic Flow	The paddle moves left or right and after colliding with minion, the minion jumps back.
Failure Condition	Player not able to move the paddle
Successful End Condition	Minion is able to move around and should be able to jump after colliding with paddle.



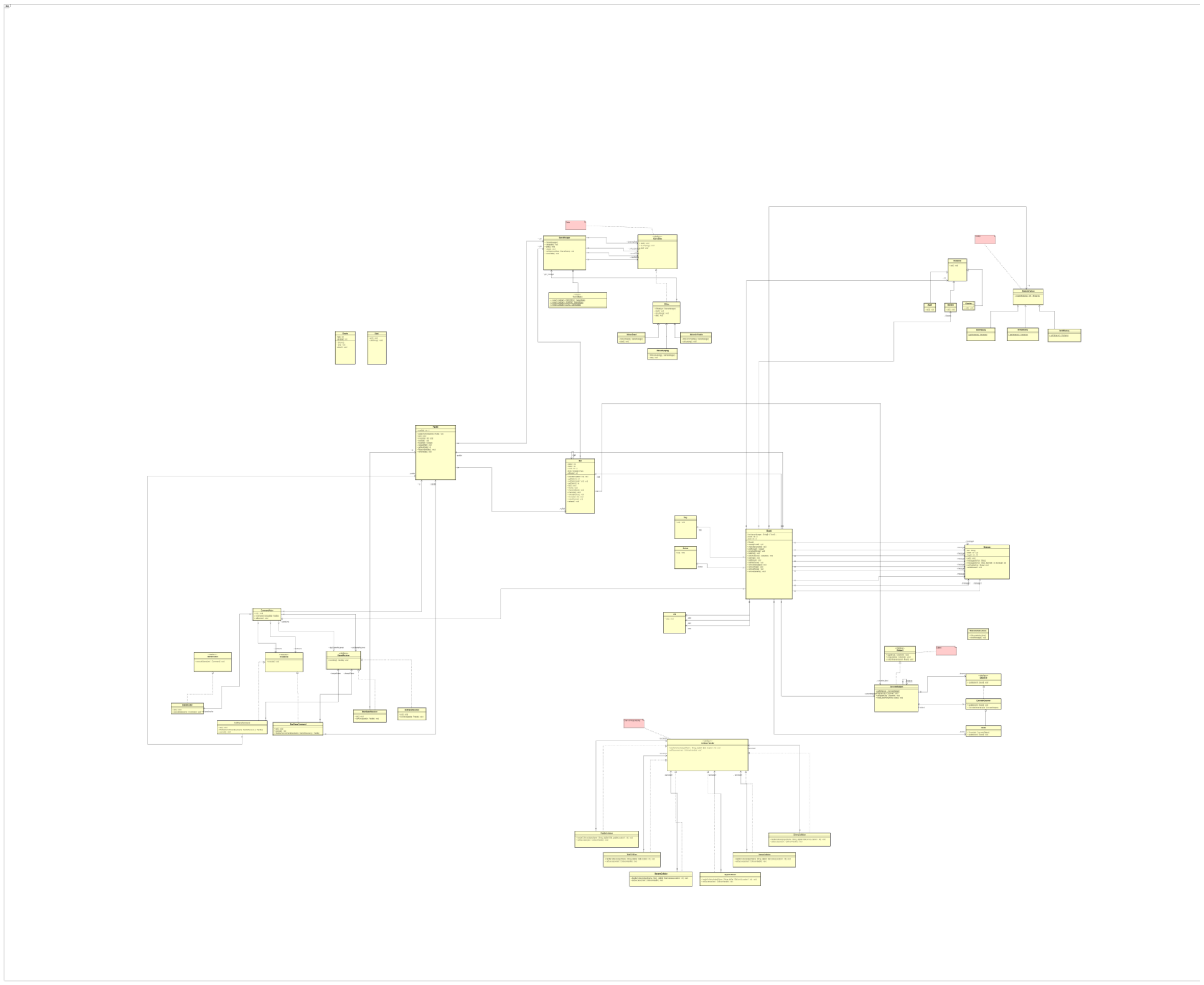
- Collect Bonus

Use Case Name	Collect Bonus
Description	The Minion can collect bonus objects when encountered to increase number of lives
Actor	Player
Precondition	Actor should be able to move around and Bonus object should be present on game screen
Trigger	Minion is jumping and bonus object is encountered.
Basic Flow	The minion collects the encountered bonus objects.
Failure Condition	The minion not able to collect the bonus object even after collision
Successful End Condition	Bonus object is collected and number of lives increases by 1.

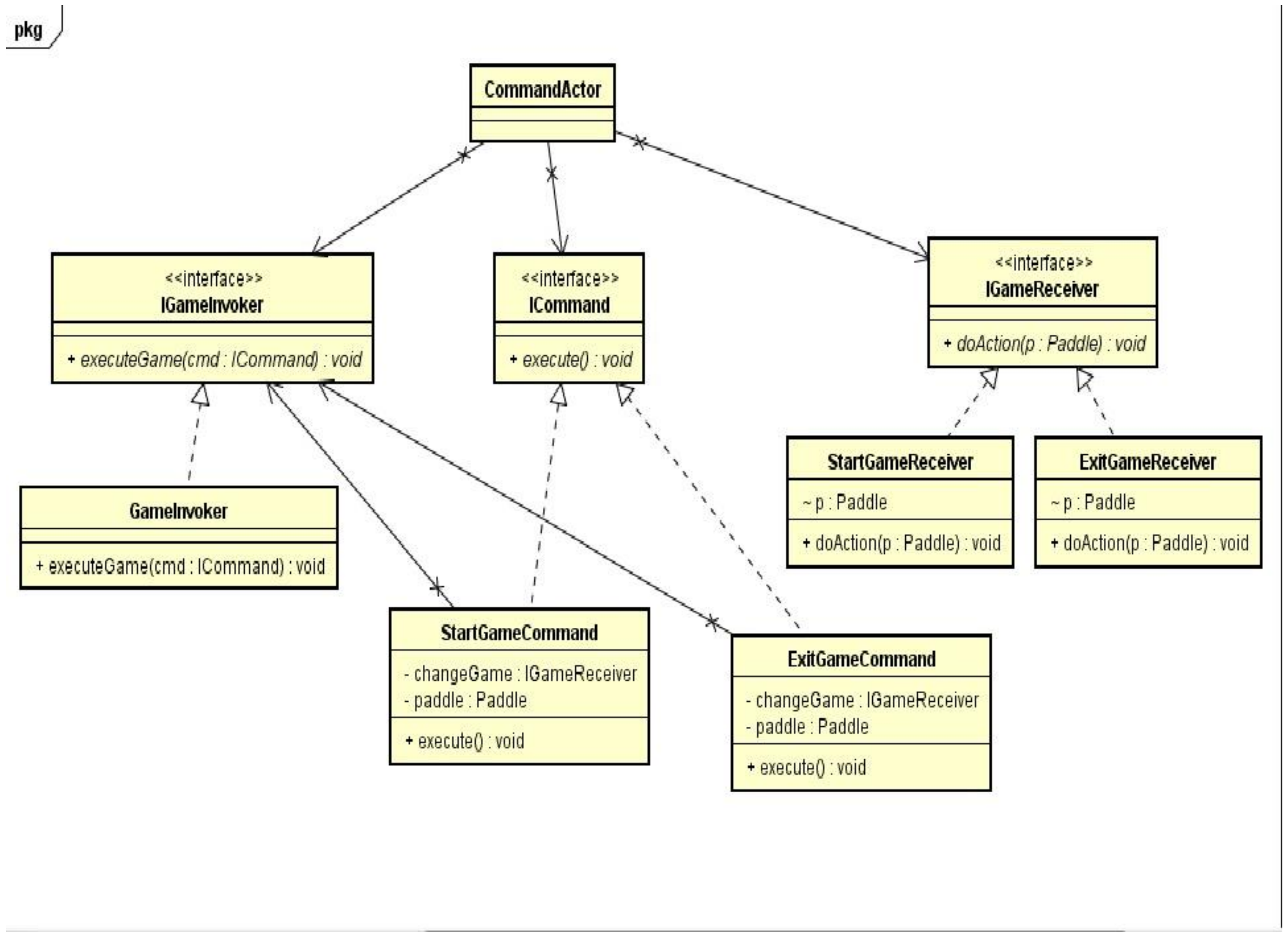


Class Design

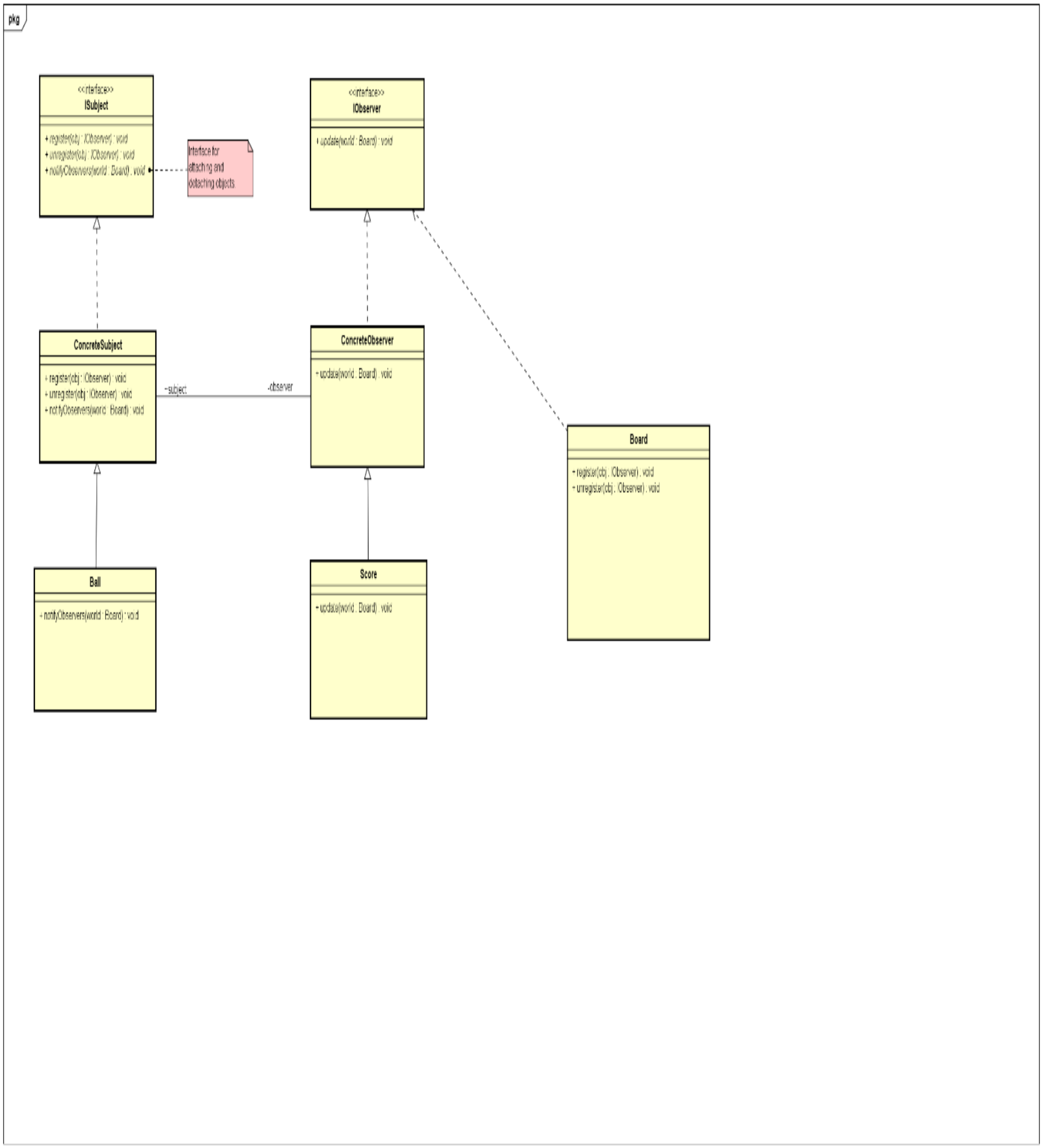
- UML Class Diagram



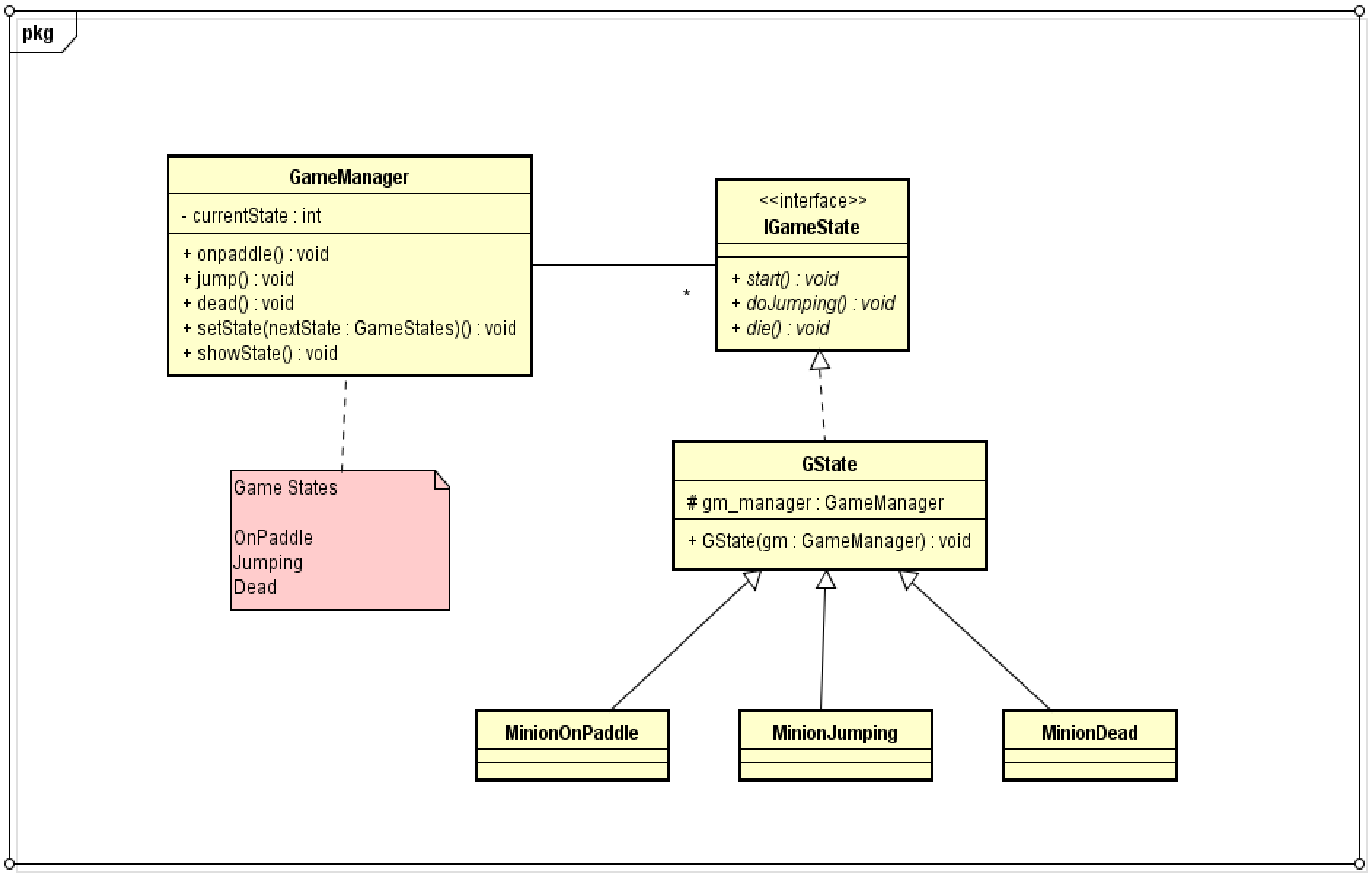
Command Pattern:



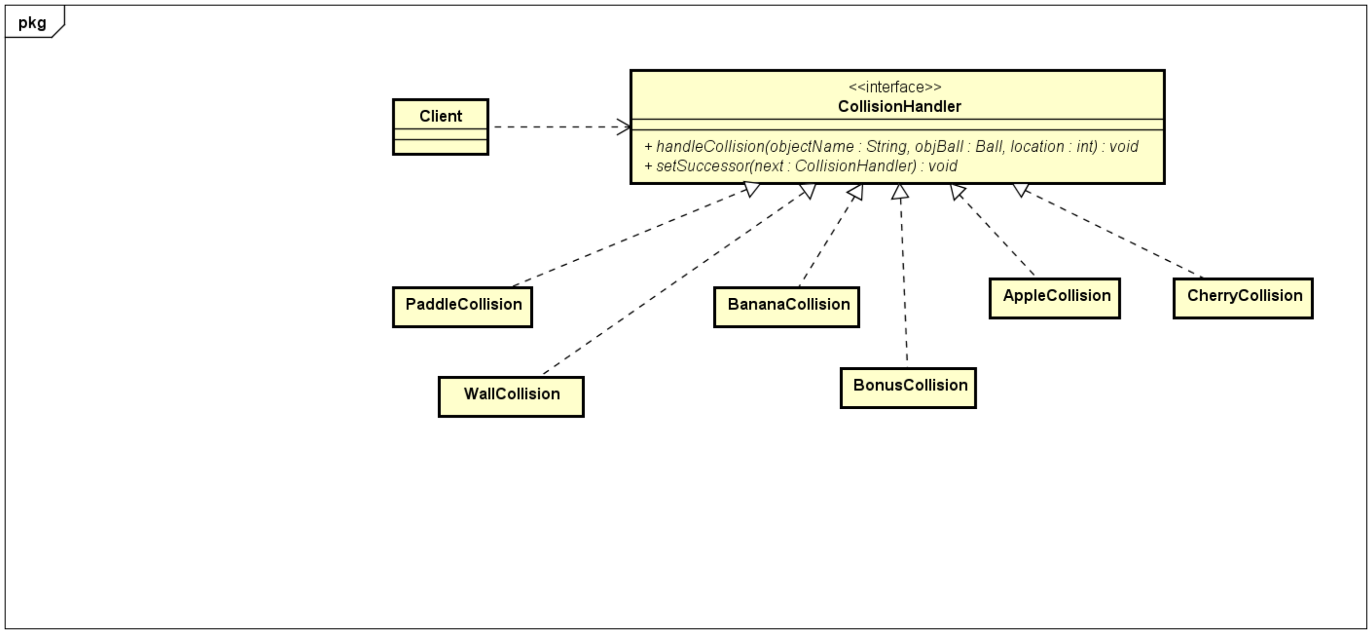
Observer Pattern:


pkg
<<interface>>
ISubject
+ register(obj : IObserver) : void
+ unregister(obj : IObserver) : void
+ notifyObservers(world : Board) : void
Interface for attaching and detaching objects
<<interface>>
IObserver
+ update(world : Board) : void
ConcreteSubject
+ register(obj : IObserver) : void
+ unregister(obj : IObserver) : void
+ notifyObservers(world : Board) : void
-subject
-observer
ConcreteObserver
+ update(world : Board) : void
Board
+ register(obj : Observer) : void
+ unregister(obj : Observer) : void
Ball
+ notifyObservers(world : Board) : void
Score
+ update(world : Board) : void

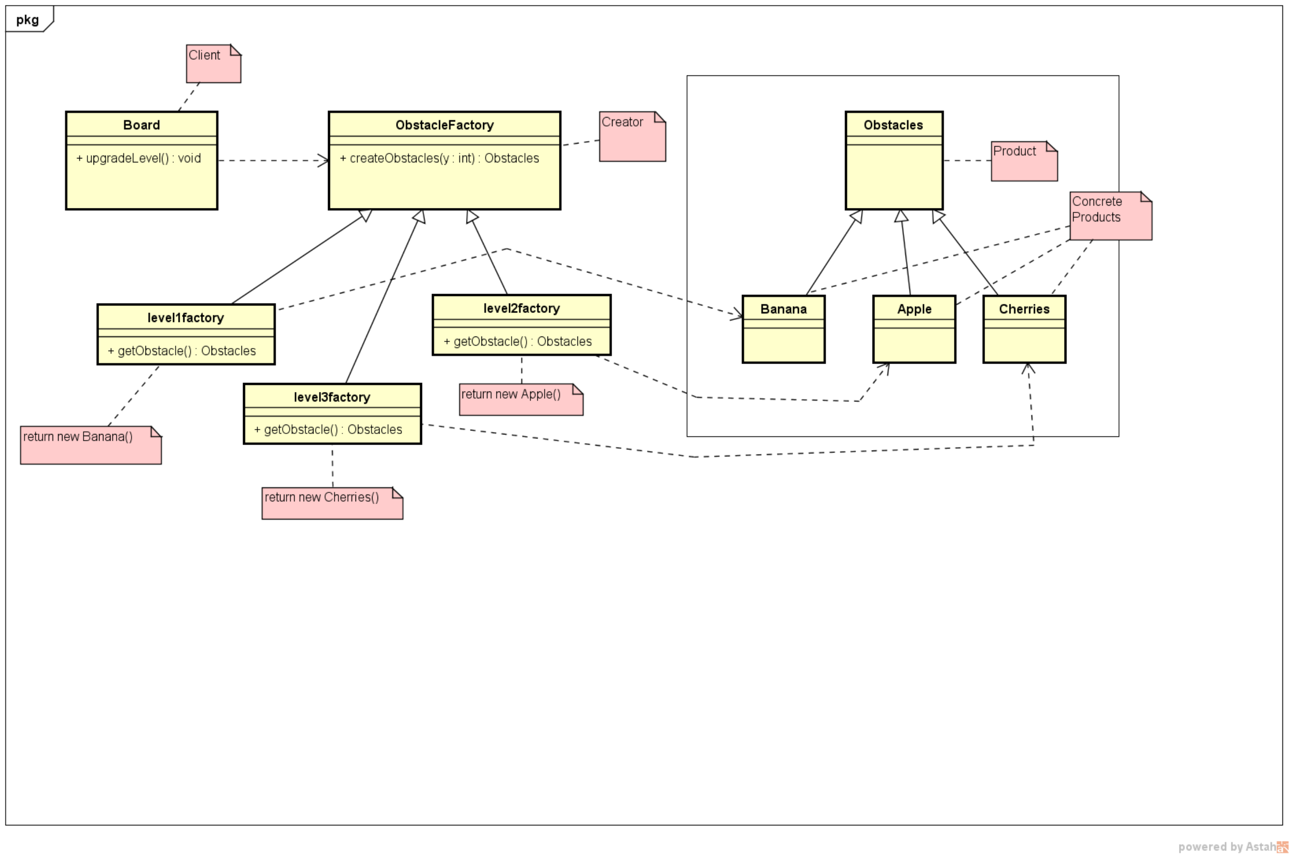
State Pattern:



Chain of Responsibility:



Factory Pattern:



YouTube Video URL:

https://www.youtube.com/watch?v=ldK4wPIM1vw