UMKC

School of Computing
and Engineering

Software Methods and Tool

Assignment 3

- Ankita Wankhede

16233344

Sequence Diagram - Asteroids Video Game

Introduction

The Sequence diagram tries to define the main logic of the game, and the whole game as the repetition of set of logic.

For better understanding and view I am trying to break the Sequence diagram in parts to explain the below sequence at the end I have published the complete sequence diagram as a whole, due to large size the image quality may get affected.

1. Initiate

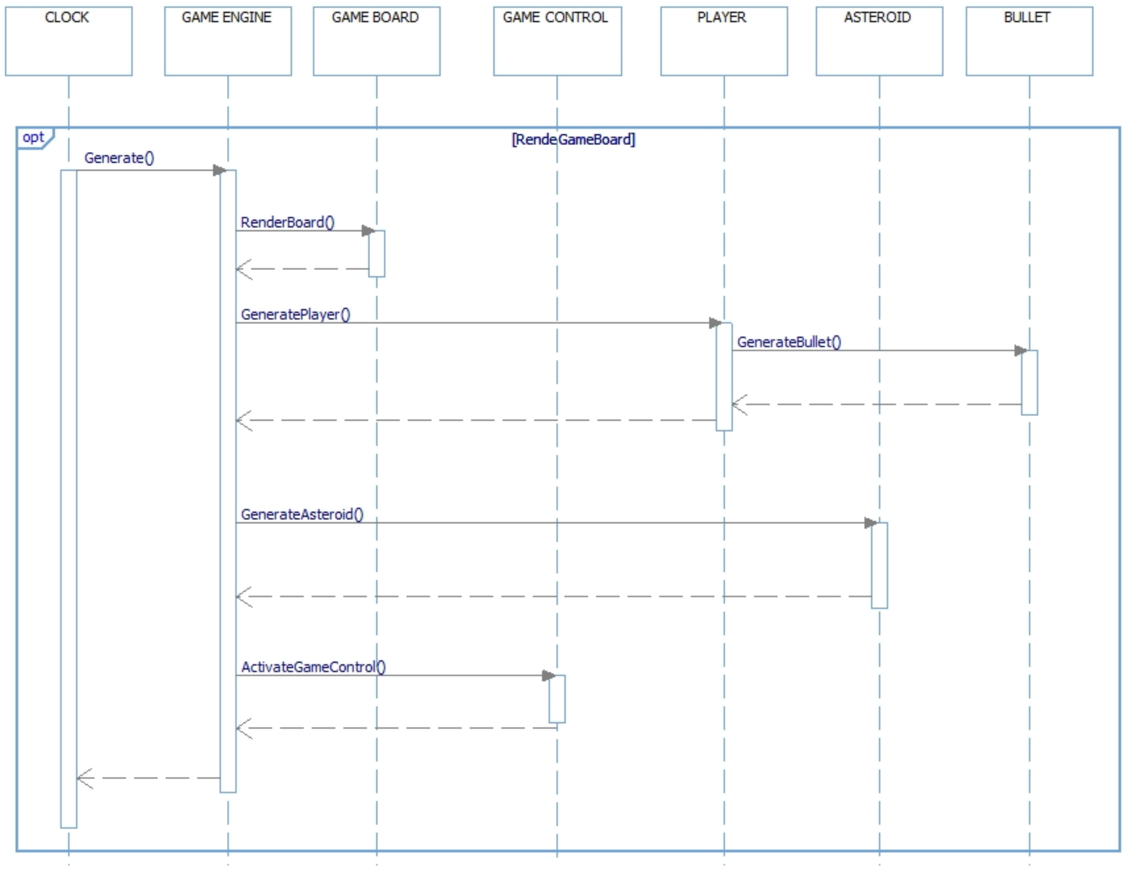
    1.1 Render game board

    1.2 Generate player (shuttle)

        1.2.1 Generate/ Load bullets

    1.3 Generate asteroids

    1.4 Activate game controls (keyboard functions thrust)



The above sequence tries to define the logic the game engine will perform at the first tick of the clock. This sequence will get executed and render game board, following which the repetitive game logic would run.

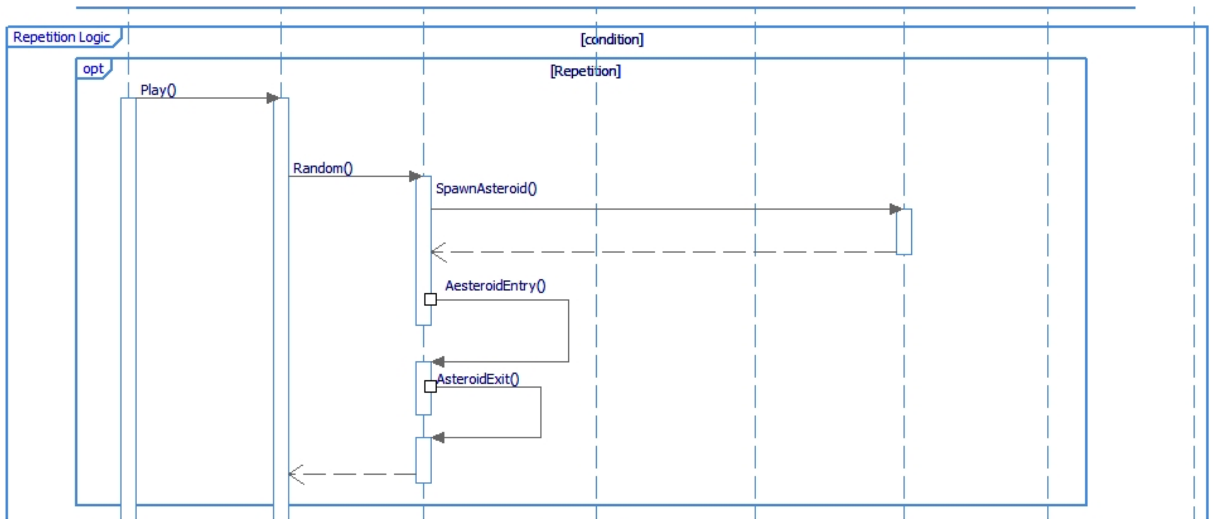
2. Play

2.1 Random position

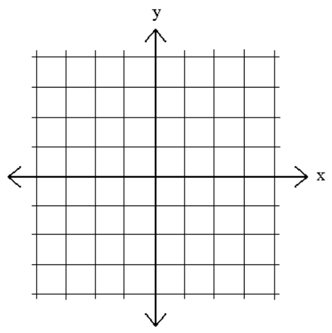
2.1.1 Spawn Asteroid (x,y location)

2.1.2 Enter Asteroid (vector)

2.1.3 Exit Asteroid (vector)



This repetition logic for asteroids explains how game engine would select random positions on the game board to spawn the asteroids, it can choose any position on the edge where the condition would be X is maximum/ minimum and Y is maximum/ minimum



the sequence diagram also represents the entry and exit of the asteroid if **Collision (False)** is satisfied

2.2 Player Action

2.2.1 Get position (initial position X=0, Y=0)

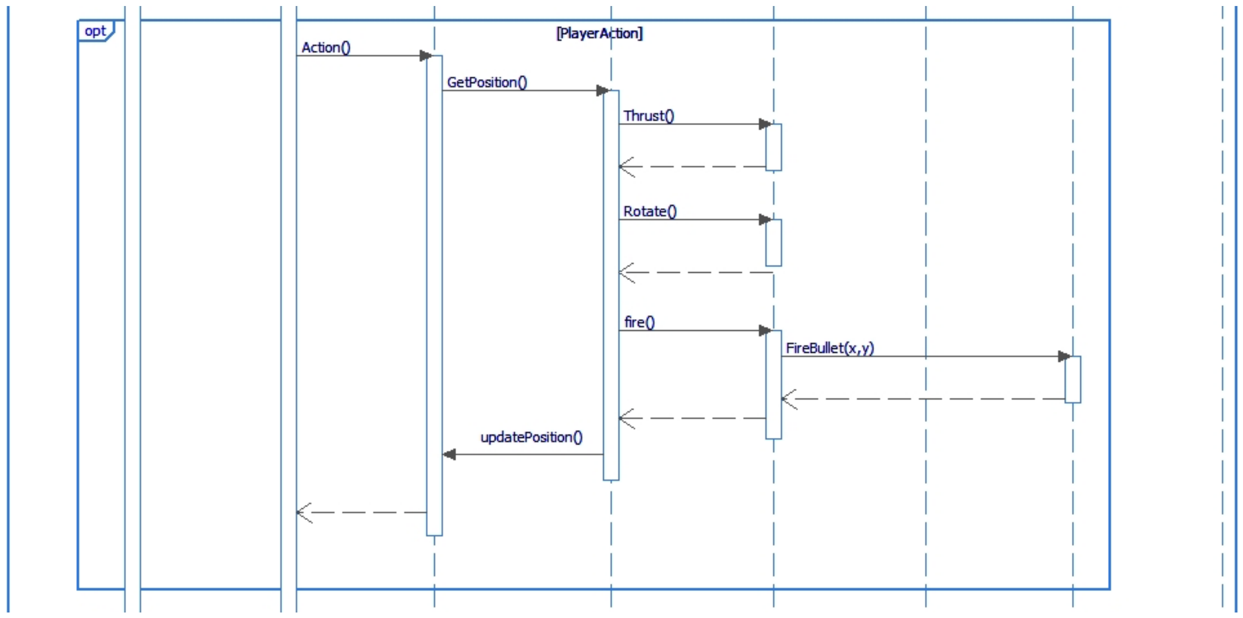
2.2.2 Thrust (vector, force)

2.2.3 Rotate (Vector, angle)

2.2.4 Fire (Game controller)

2.2.4.1 Firebullet (Palyerposition, bulletvector)

2.2.5 update position (New Position x, y)



Player action sequence tries to represent the logic where the game engine will constantly keep track of the Player position and actions need to be performed as part of game controller like Thrust, change direction, fire bullet and store the updated position in order to decide the collision logic.

2.3 Collision

2.3.1 Get Player Position

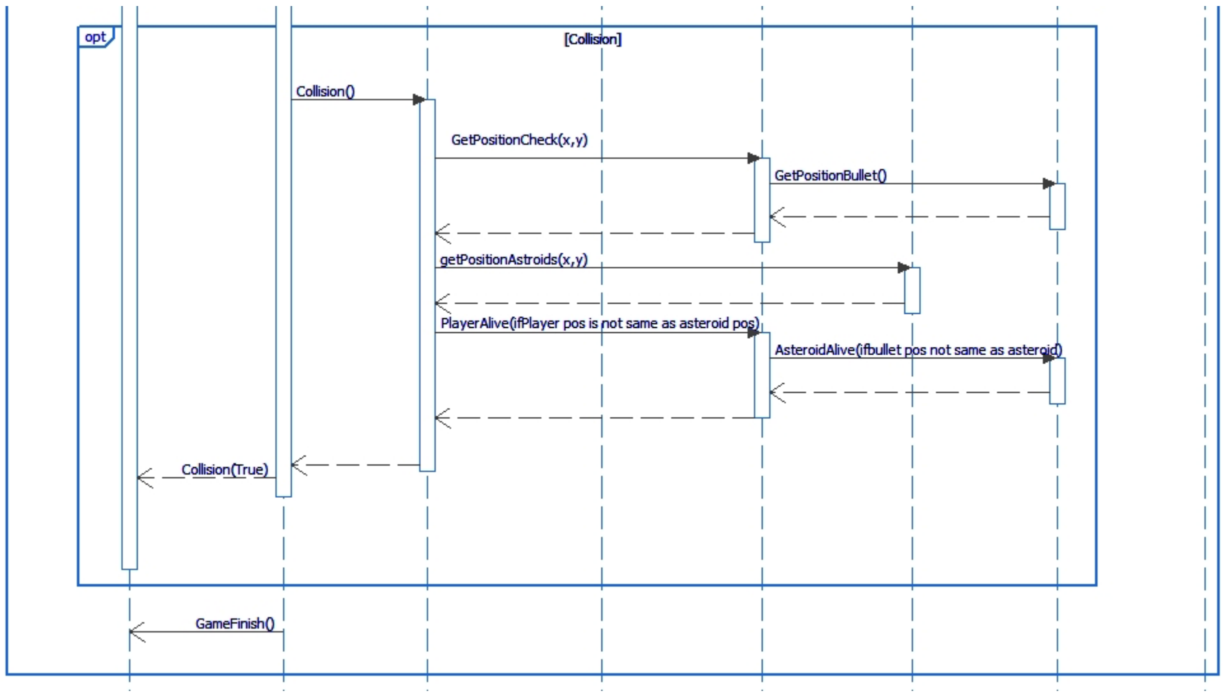
2.3.1.1 Get Bullet Position

2.3.2 Get Asteroid Position

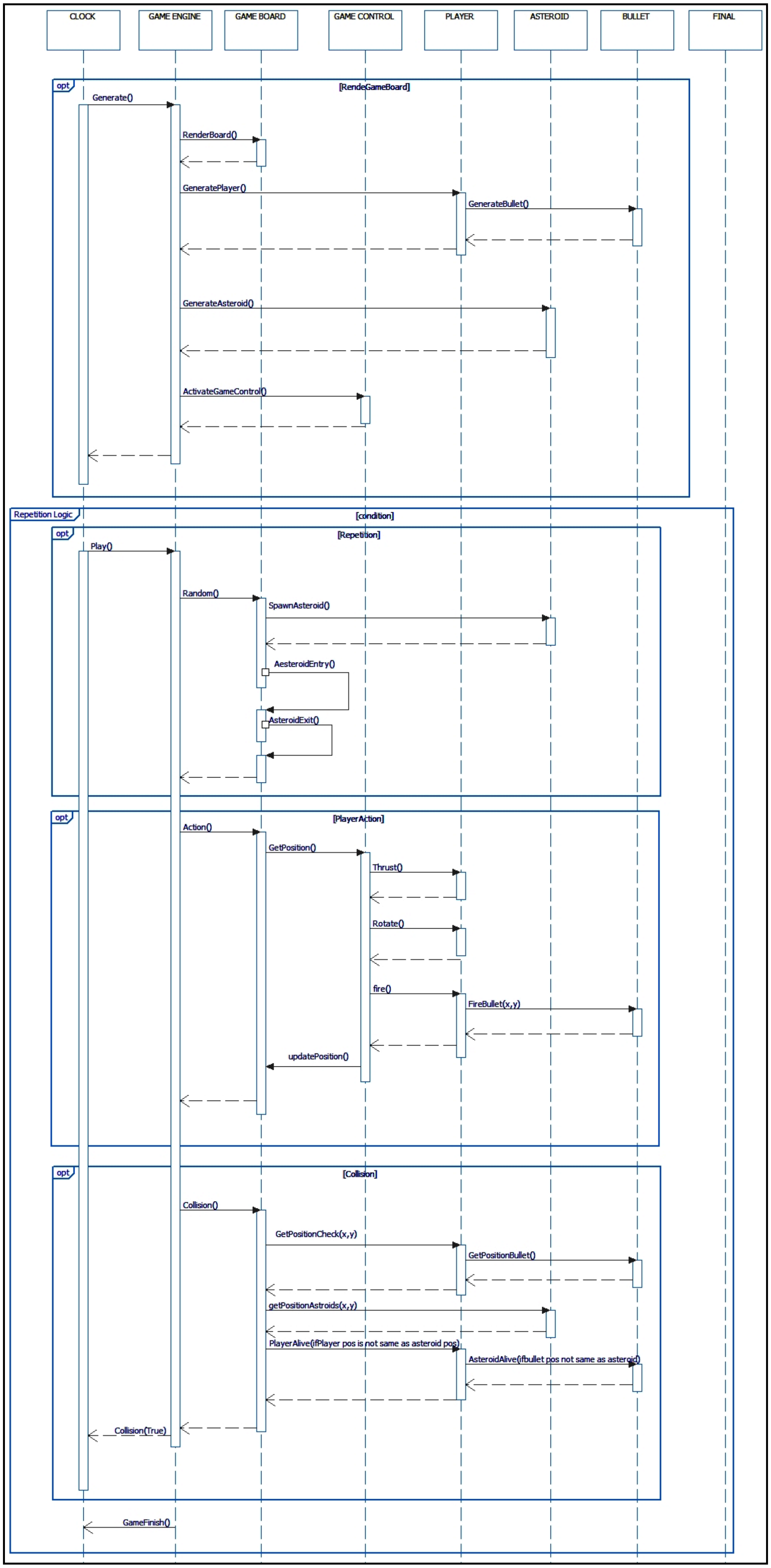
2.3.3 Player Alive

2.3.4 Asteroid Alive

2.3.5 Collision(True)



The collision sequence shows how the game engine will constantly try to get the position of the Asteroid, bullet and Player and try to interpose them in order to determine if a collision has occurred. if the Bullet positions interposes with the asteroid position it will determine the asteroid has been hit by the bullet, and if the asteroid position is interposed with player, the game engine would be able to determine that a collision has occurred and life must be reduced or stop game if value of life=0.


CLOCK
GAME ENGINE
GAME BOARD
GAME CONTROL
PLAYER
ASTEROID
BULLET
FINAL
opt
[RendeGameBoard]
Generate()
RenderBoard()
GeneratePlayer()
GenerateBullet()
GenerateAsteroid()
ActivateGameControl()
Repetition Logic
[condition]
opt
[Repetition]
Play()
Random()
SpawnAsteroid()
AesteroidEntry()
AsteroidExit()
opt
[PlayerAction]
Action()
GetPosition()
Thrust()
Rotate()
fire()
FireBullet(x,y)
updatePosition()
opt
[Collision]
Collision()
GetPositionCheck(x,y)
GetPositionBullet()
getPositionAstroids(x,y)
PlayerAlive(ifPlayer pos is not same as asteroid pos)
AsteroidAlive(ifbullet pos not same as asteroid)
Collision(True)
GameFinish()

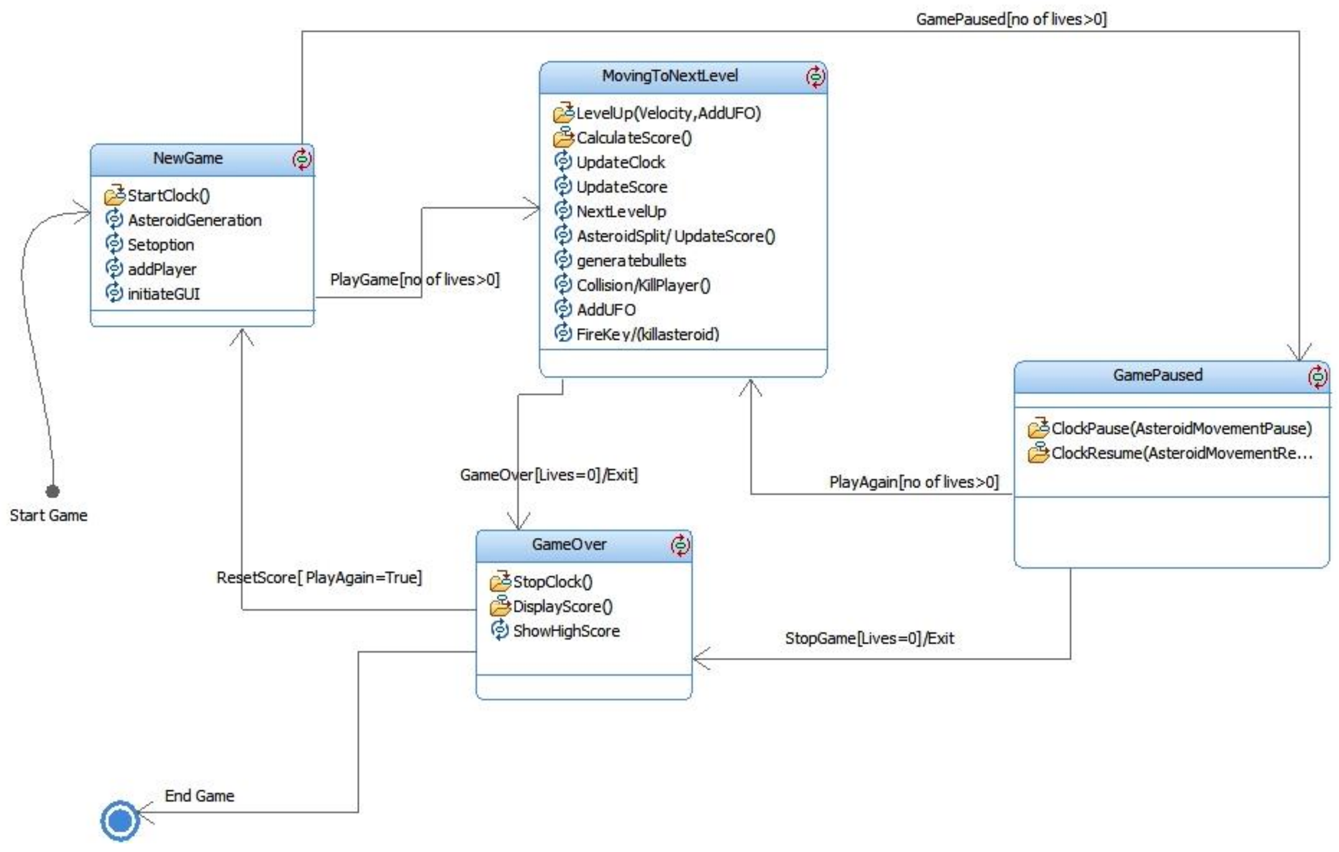
State Diagram

Introduction

State diagram describes the life cycle of an object in a class. they show all possible states the object can get into and transitions how the states change.

Below state diagram for Asteroid game tries to elaborate the states for game control class. The game controller state diagram handles the different states within the game, which will be:

- NewGame
- MovingtoNextLevel
- GamePaused
- GameOver



- Within the **NewGame** state, the game's default screen will load, bearing the game title, after which the game play will begin.
- During the game play based on the player's performance the player transitions to **MovingToNextLevel** state, in order to increase the difficulty and challenge the player.
- Within the **GamePaused** state the game clock will be suspended in steady state and all game features and GUI will stay steady, until the game is unpaused, during which the clock will pick up from the last state the game was left in.
- **GameOver** state, the game will display the score obtained so far, wait for Player, if the player chooses to restart the game, the game will simply reset to NewGame state.