Virtual Arcade

Requirements and Analysis Document (RAD)

Version: 3.0
Date: 23/10 - 2020
Authors: *Georges Kayembe, Mhd Jamal Basal, Jonathan Stigson, Oliver Ljung, Robert Sahlqvist*

1 Introduction

1.1 Purpose of application

The project aims to implement a virtual arcade with five or more integrated retro games. The application has great focus on recreating a nostalgic feeling whilst introducing a modern vibe into old retro arcade games.

1.2. General characteristics of application

The program is a platform independent desktop application able to launch five different retro games. The five games available are as follows: Breakout, Frogger, Pong, Snake and Space Invaders.

1.3 Definitions, acronyms, and abbreviations

GUI: “Graphical User Interface” also “User Interface”, the part of the program that the user sees and interacts with.

DoD: “Definition of Data”, a list of requirements that have to met for a User Story to be considered done.

User Story: short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

RAD: “Requirement and Analysis document”.

SDD: “System Design Document”.

UML: “Unified Modeling Language”, A visual language for mapping and representing systems. Can be used for modelling och class diagrams, sequence diagrams and more. In this project used for Domain and design model.

2 Requirements

2.1 User Stories

Story Identifier: GM
Story Name: Game Selection
Description: As a player, I want to be able to select a game from the game selection menu and so that I can then play it.
Confirmation: Story is done the player is able to select a game and then be able to play said game.
Functional: When the player is in the game selection menu he/she should be able to select a game and then be taken to that game,

Story Identifier: P1
Story Name: Pong Movement
Description: As a player, I want to be able to move either the right or the left paddle, depending on if I'm player one or player two, so that I am able to stop the ball from passing my paddle.
Confirmation: Story is done when the paddle or paddles move in the correct direction when the player presses the arrow keys.
Functional: If the player presses the up arrow key, the paddle moves up. If the player presses the down arrow key, the paddle moves down.

Story Identifier: P2
Story Name: Pong Score
Description: As a player, I want to be able to receive points when I score a goal.
Confirmation: Story is done when a player gets a point if said player scores a goal on the opposite player's paddle.
Functional: If the ball goes past a paddle, the player with the opposite paddle gets a point on the scoreboard at the top of the screen. The scoreboard numbers are placed on either side of a dividing line.

Story Identifier: P3
Story Name: Pong Ball Bounce (Players)
Description: As a player, I want the ball to bounce off the players.
Confirmation: Story is done when the ball bounces off the players' paddles if the ball hits them.
Functional: If the ball "touches" any of the paddles, the ball's x direction is flipped so that it goes in the opposite direction.

Story Identifier: P4

Story Name: Pong Ball Bounce (Walls)
Description: As a player, I want the ball to bounce off the walls.
Confirmation: Story is done when the ball bounces off the top and bottom walls if the ball hits any of them.
Functional: If the ball "touches" the top or the bottom wall, the ball's y direction is flipped so that it goes in the opposite direction.

Story Identifier: P5
Story Name: Pong Game Pause
Description: As a player, I want to be able to pause the game.
Confirmation: Story is done when a player is able to pause the game with the push of a button.
Functional: If the player hits a designated key, everything in the game stops moving and becomes unresponsive until the player hits the key again.

Story Identifier: P6
Story Name: Pong AI Player
Description: As a player, I want to be able play against an AI bot if a second player isn't available.
Confirmation: Story is done when a player is able to play against a bot rather than another player.
Functional: If the player hits a designated key, then the right paddle becomes controlled by a bot which reacts to the ball's position.

Story Identifier: P7
Story Name: Pong Ball Respawn
Description: As a player, I want the ball to respawn in the middle after a goal.
Confirmation: Story is done when the ball disappears and then reappears in the middle of the game after it scores a goal.
Functional: If the ball hits the right or left edge of the screen it despawns and respawns in the middle of the game with a random direction.

Story Identifier: SI1
Story Name: SpaceInvaders Player Movement
Description: As a player, I want to be able to move the player using the arrow keys so that I can avoid getting hit by enemies.
Confirmation: Story is done when the player character moves in the correct direction when the player presses the arrow keys.
Functional: If the player presses the right arrow key, player moves right. If player presses the left arrow key, player moves to the left

Story Identifier: SI2

Story Name: SpaceInvaders Shooting
Description: As a player, I want to be able to shoot when I press a button and with that shot I want to be able to destroy enemy aliens so that I can get rid of all the aliens.
Confirmation: Story is done when the player shoots a projectile that can destroy enemy aliens with the press of a specific button.
Functional: If the player presses the "shoot button"(button not determined yet) a projectile that can destroy enemy aliens will be spawned.

Story Identifier: SI3
Story Name: SpaceInvaders Score
Description: As a player, I want to be able to gain score by destroying enemy aliens, so that I can get a better high score.
Confirmation: Story is done when the player gains score by destroying enemy aliens.
Functional: If the player shoots and destroys an enemy alien score should be added.

Story Identifier: SI4
Story Name: SpaceInvaders Death and Respawn
Description: As a player, I need to be able to die and then respawn if I have lives left so that the game will be challenging.
Confirmation: Story is done when the player dies upon collision with an alien bullet, player also has 3 lives.
Functional: If the player collides with an enemy bullet he/she will die and respawn if there are lives left.

Story Identifier: SI5
Story Name: SpaceInvaders Enemy Movement
Description: As a player, I want the enemy aliens to move in the original pattern so that the game is recognizable.
Confirmation: Story is done when the player can visually see that the enemies are moving in the original games pattern.
Functional: When the game updates the enemies shall move according to the original pattern

Story Identifier: SI6
Story Name: SpaceInvaders Pause
Description: As a player, I want to be able to pause the game so I can take a breather if needed.
Confirmation: Story is done when the player can pause the game and with that get a pause menu.
Functional: When the player presses the escape button the game will pause and the player will arrive at a 'pause menu'.

Story Identifier: F1

Story Name: Hippity hoppity
Description: As a user, I need to be able to move the Frog so that I can complete the goal of the game.
Confirmation: User Story is done when keypresses register and frog moves accordingly.
Functional: Does the frog move when I use the arrow keys? Does the frog stay within the boundaries of the game's playing field?

Story Identifier: F2
Story Name: Lives lost
Description: As a user, I need to know what the rules of interacting with obstacles are so I have the necessary information to complete the game.
Confirmation: User Story is done when clear indicators (visual or other) answer questions of functional criteria and align with functionality of the program.
Functional:
Do I have multiple lives?
Do I lose a life when I get hit by a car?
Do I lose a life if I fall into the river?

Story Identifier: F3
Story Name: End of the road
Description: As a user, I need to know what the rules of winning/losing are so I have the necessary information to complete the game.
Confirmation: User Story is done when clear indicators (visual or other) answer questions of functional criteria and align with functionality of the program.
Functional:
Do I get "Game Over" if I lose all my lives?
Do I get points for getting a frog across the road and the river?
Do I finish the level if I get all the frogs across the road and the river?

Story Identifier: F4
Story Name: Modern Vibe
Description: As a user, I need the game to look new and modern while still maintaining the feeling of the original game so that I can relive what it was like to play the original while still feeling like it is a new and modern game.
Confirmation: User Story is done when functionality and visuals match the original Frogger (good enough).
Non-functional: Is the game recognisable?

Story Identifier: SG1
Story Name: Snake Player Movement
Description: As a player, I want to be able to move the player using the arrow keys so that I can turn towards the food.

Confirmation: Story is done when the player character moves in all four directions when the player presses the arrow keys.
Functional: Does the Snake move when I use the arrow keys?

Story Identifier: SG2
Story Name: Snake Score
Description: As a player, I want to be able to gain score by eating food, so that I can get a better high score.
Confirmation: Story is done when the player gains score by eating food.
Functional: If the player eats a food score should be added.

Story Identifier: SG3
Story Name: Snake Death and Restart
Description: As a player, I need to be able to die and then start over.
Confirmation: Story is done when the player dies upon collision with himself or a wall.
Functional: If the player collides with an himself/wall he will die and start over again .

Story Identifier: SG4
Story Name: Snake Pause
Description: As a player, I want to be able to pause the game so I can take a breather if needed and also go back to the menu.
Confirmation: Story is done when the player can pause the game and with that get a pause menu.
Functional: When the player presses the escape button the game will pause and the player will arrive at a ‘pause menu’.

Story Identifier: SG5
Story Name: Snake Speed
Description: As a player, I need to be able to choose the player's speed (Slow, normal, fast and insane).
Confirmation: Story is done when the player chooses one of these speeds.
Functional: The player's speed will be increased according to the speed chosen.

Story Identifier: SG6
Story Name: Wall Collisions
Description: As a player, I need to be able to choose between activating/deactivating a collision with the wall.
Confirmation: Story is done when the player chooses Off/On (Wall Collisions) .
Functional: If the Wall Collisions (OFF): The player appears form the opposite side when the collision occurs. If the Wall Collisions (ON): The player will die when the collision occurs.

Story Identifier: SG7
Story Name: Gameboard Size
Description: As a player, I need to be able to choose the Gameboard size (Small, medium, and big).
Confirmation: Story is done when the player chooses one of these sizes.
Functional: The board size will be changed according to the size chosen.

Story Identifier: BO1
Story Name: Breakout Menu1
Description: As a player, I want to be able to see a description about the game to understand how the game works.
Confirmation: Story is done when the player starts the game and clicks on the button “Help” from the game menu.
Functional: If the player clicks on the ”Help button”, the player will see a short description about the game.

Story Identifier: BO2
Story Name: Breakout Menu2
Description: As a player, I want to be able to see a record of the best score for the purpose of measuring upp myself and increasing my skills .
Confirmation: Story is done when the player starts the game and clicks on the button “Scores” from the game menu.
Functional: If the player clicks on the ”Score button”, the player will see a list which contains the five best players and high scores.

Story Identifier: BO3
Story Name: Breakout Menu3
Description: As a player, I want to be able to identify myself in order to track my score.
Confirmation: Story is done when the player starts the game and clicks on the button “Play” from the game menu. Then the player will be able to write his first & last name and start a new game.
Functional: If the player presses the ”Play button”, the player will be able to identify his/her self and then starts a new game.

Story Identifier: BO4
Story Name: Breakout Stop
Description: As a player, I want to be able to end upp the current breakout game so that I can choose to start a new one.
Confirmation: Story is done when the player clicks on EXIT-button.
Functional: If the player presses the ”EXIT button”, the player will be able to move to the game menu and start a new game.

Story Identifier: BO5
Story Name: Breakout Pause
Description: As a player, I would like to be able to pause the game-play, in order to return to the game at a later time.
Confirmation: Story is done when the player clicks on PAUSE. The game will continue when the player clicks on PAUSE again.
Functional: If the player presses the "PAUSE button" the game will pause successfully. When the player hits the "PAUSE button" again the game will resume as usually.

Story Identifier: BO6
Story Name: Breakout Move
Description: As a player, I would like to be able to move the paddle horizontally (right and left) for the purpose to hit the ball.
Confirmation: Story is done when the player clicks on 'Q' or 'W'.
Functional: If the player presses the key 'Q' the paddle would move to the left. If the player presses the key 'W' the paddle would move to the right.

Story Identifier: BO7
Story Name: Breakout Score1
Description: As a player, I would like to see my current score while playing the game.
Confirmation: Story is done when the player starts a new game.
Functional: The score will be displayed at the top of the screen while the player is playing the game.

Story Identifier: BO8
Story Name: Breakout Score2
Description: As a player, I would like to save my score when I lose or win a game.
Confirmation: Story is done when the player hits the last brick or when the ball hits the bottom. The score will automatically be saved.
Functional: If the player hits the last brick or if the ball hits the bottom, the score will automatically be saved. The player should be able to see his/her score if and only if the score is in the top 5 high scores.

2.2 Definition of Done

To determine whether a User Story has been completed or not, all criteria from either group of criteria need to be fulfilled. Those groups of criteria are:
Group 1:

- If there can visually or otherwise be determined that it fulfills its intended purpose.
- If it can fully be tested via JUnit.

Group 2:

- If there can visually or otherwise be determined that it fulfills its intended purpose.
- If peer review determines that it fulfills its intended purpose.

Group 2 is only applicable if the aspects of the User Story cannot be tested via JUnit.

2.3 User interface

Screenshots from the application:

Figure 1. Start Menu:

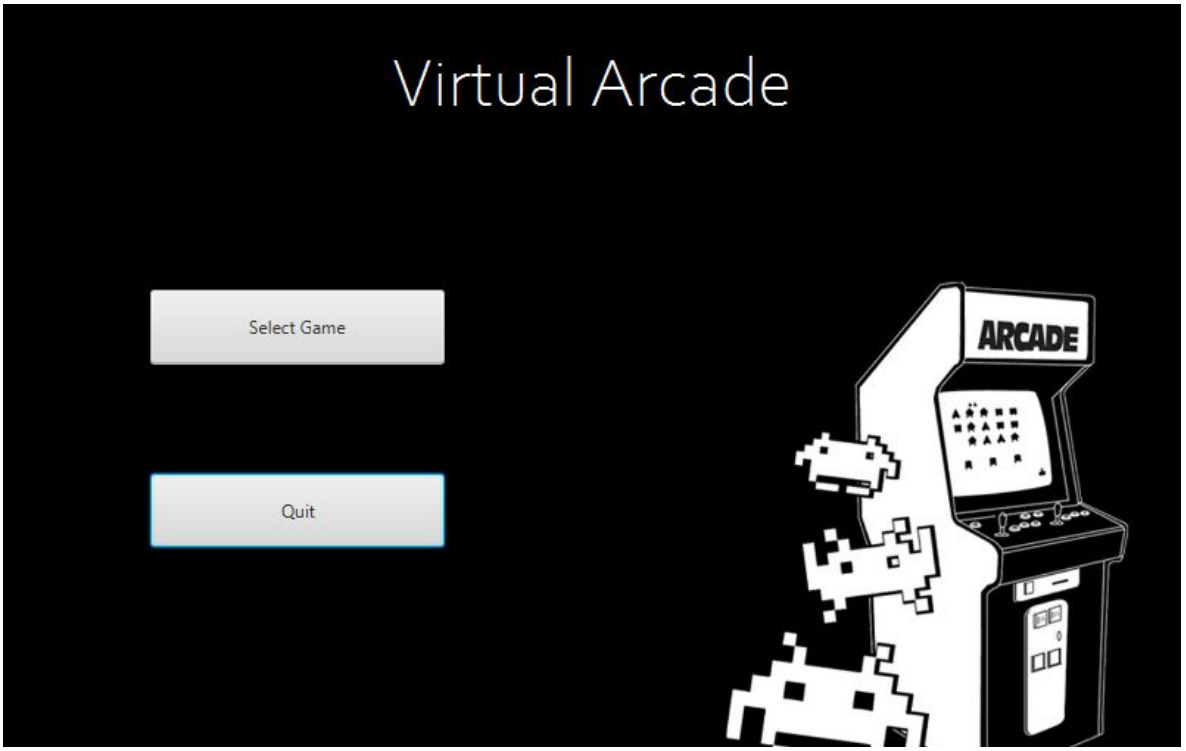


Figure 2. Game Select Menu:

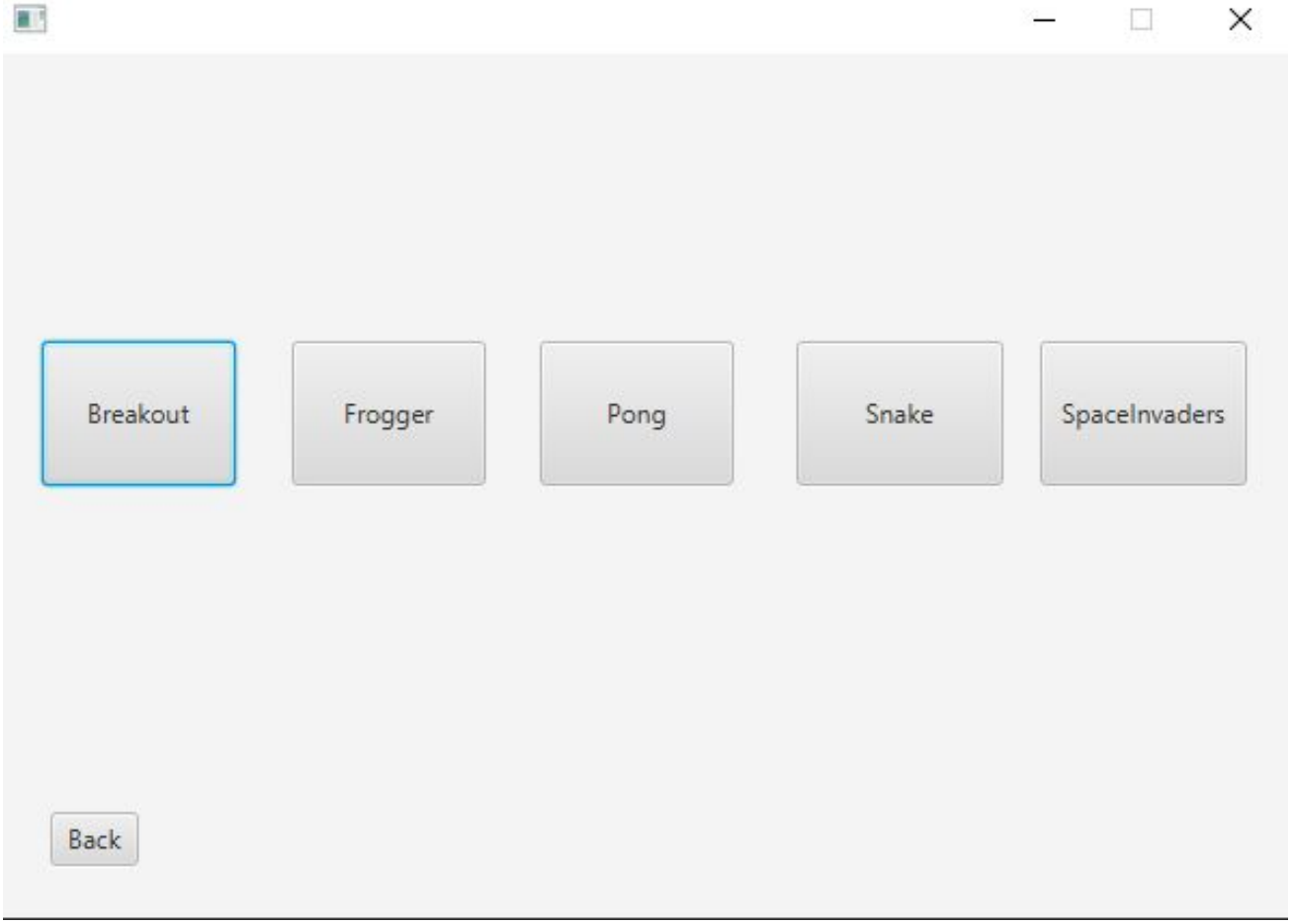


Figure 3. Breakout Menu:

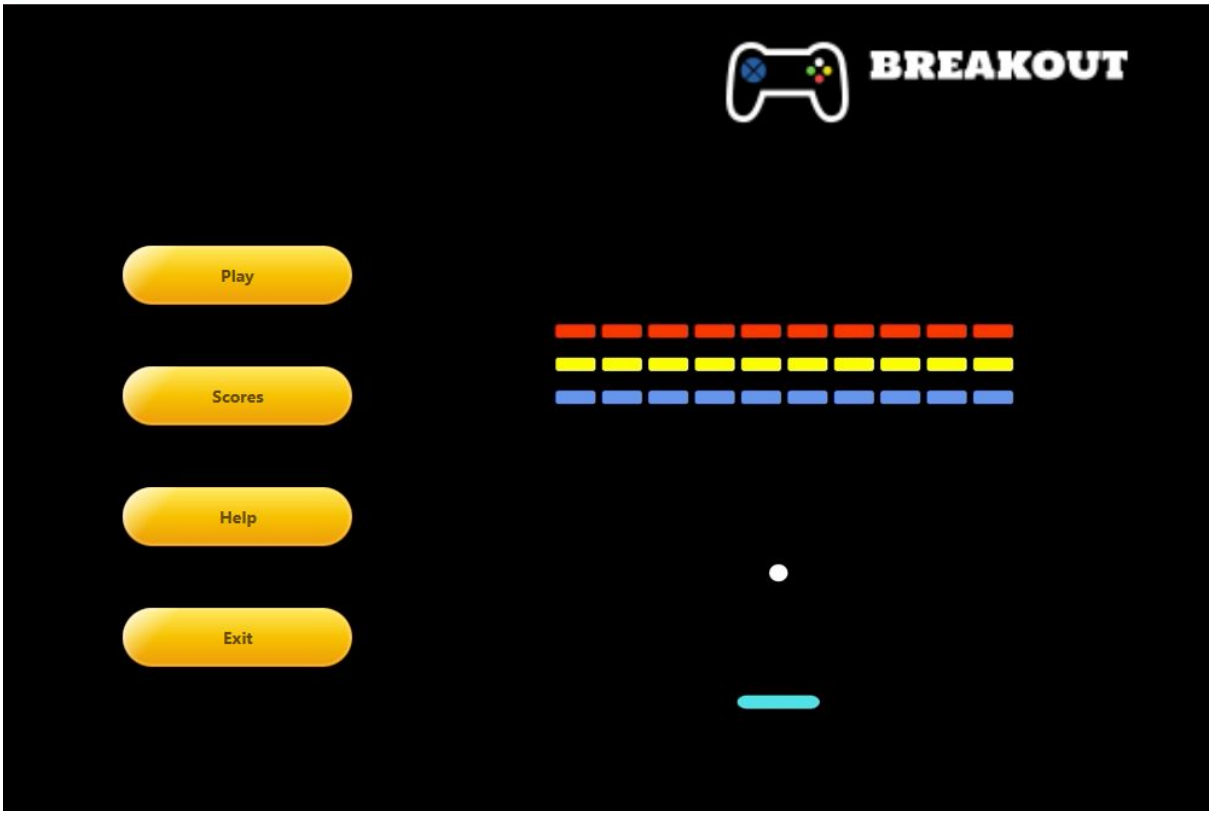


Figure 4. Breakout Game:

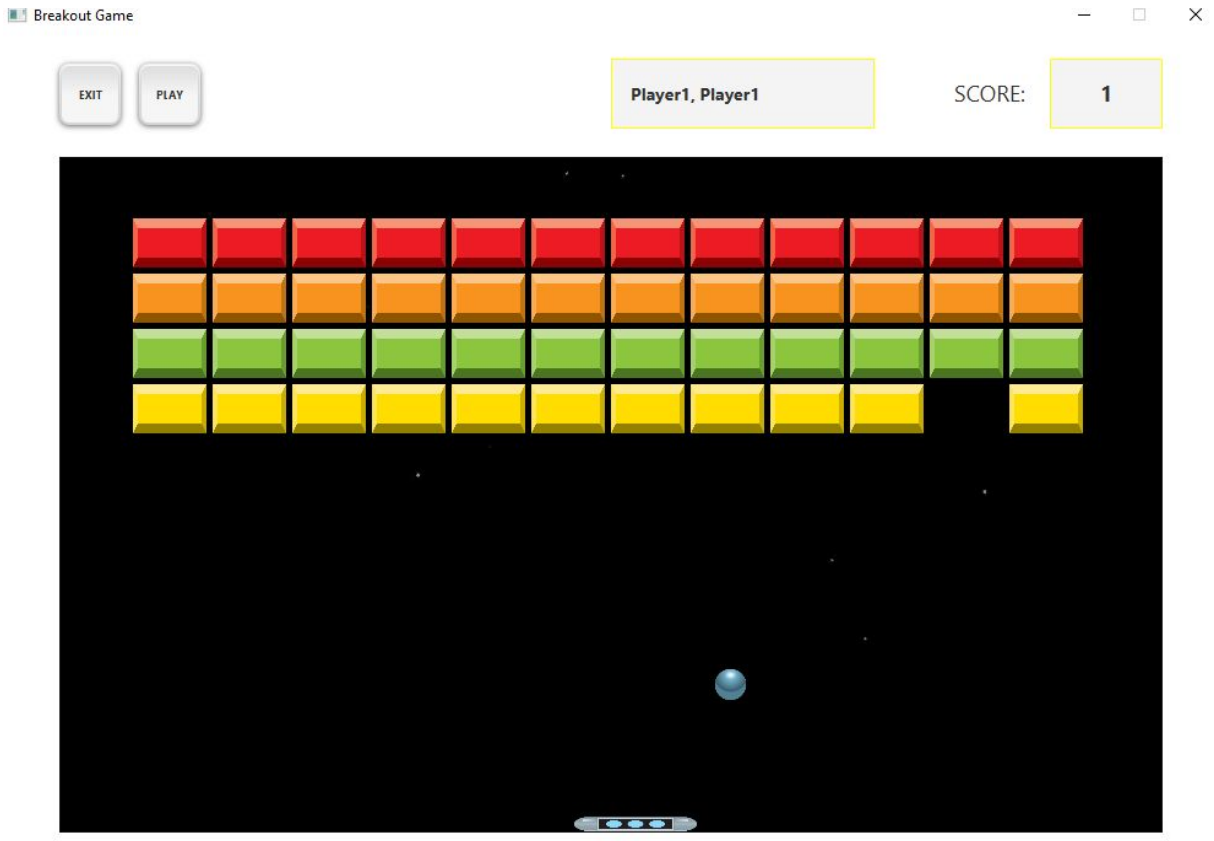


Figure 5. Frogger Game:

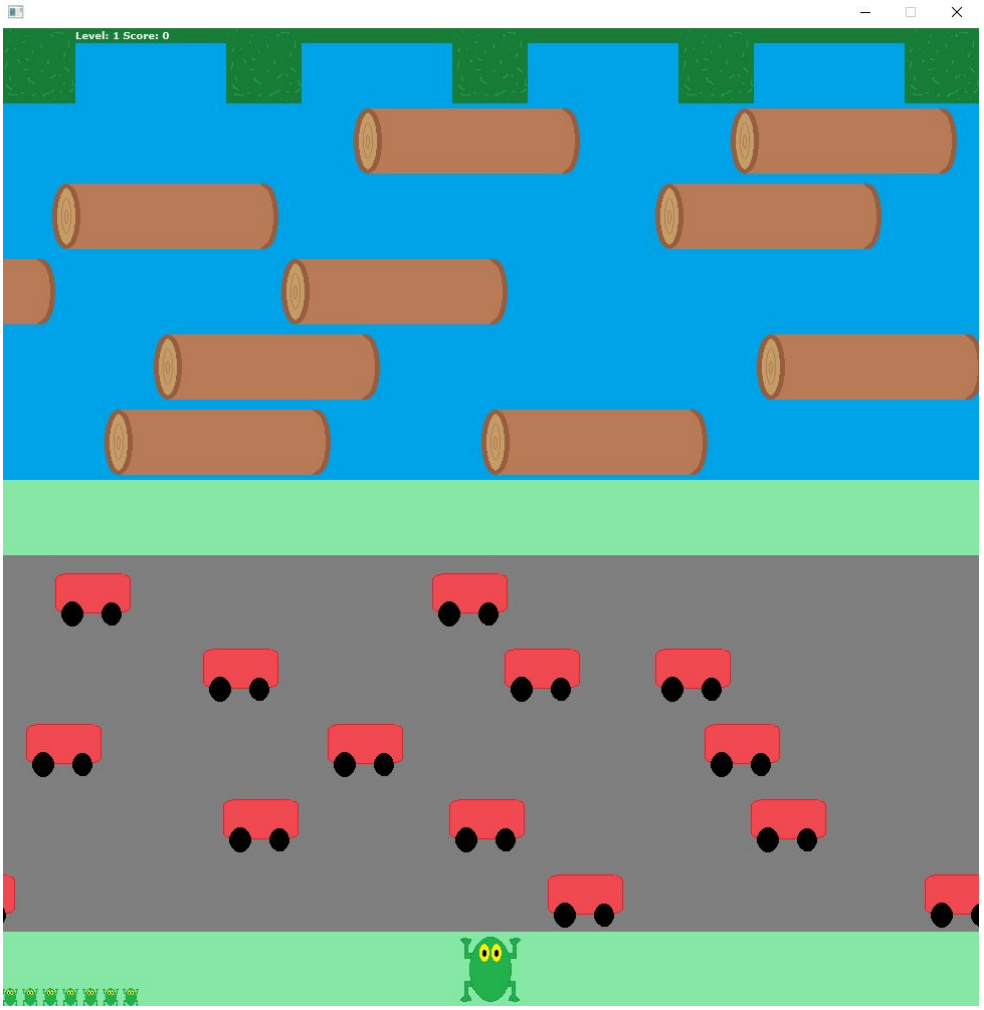


Figure 6. Pong Game:

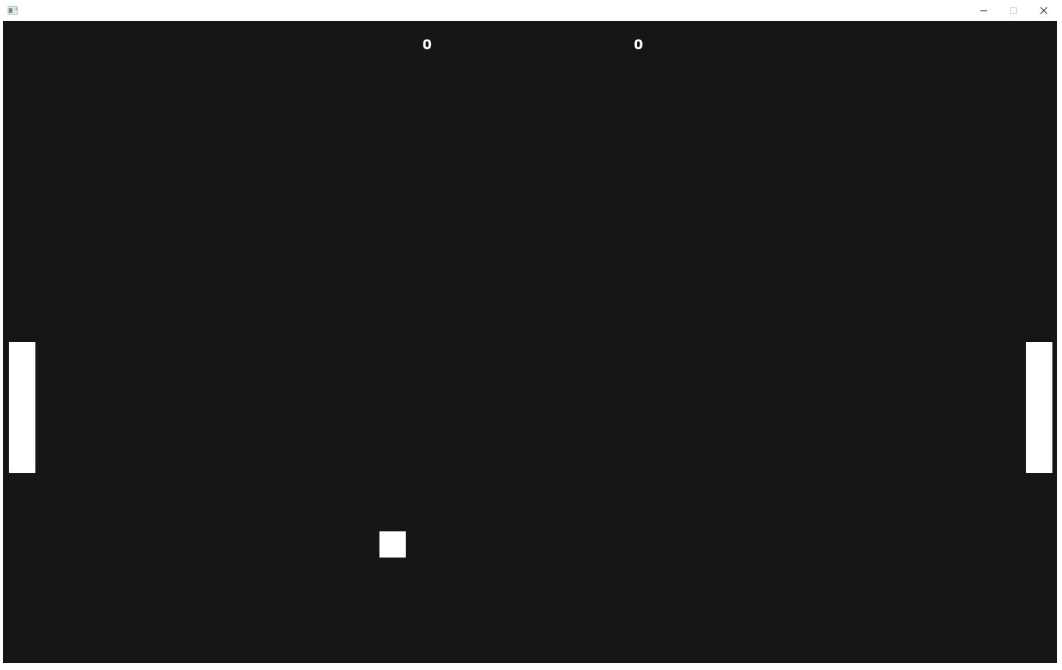


Figure 7. Snake Menu:

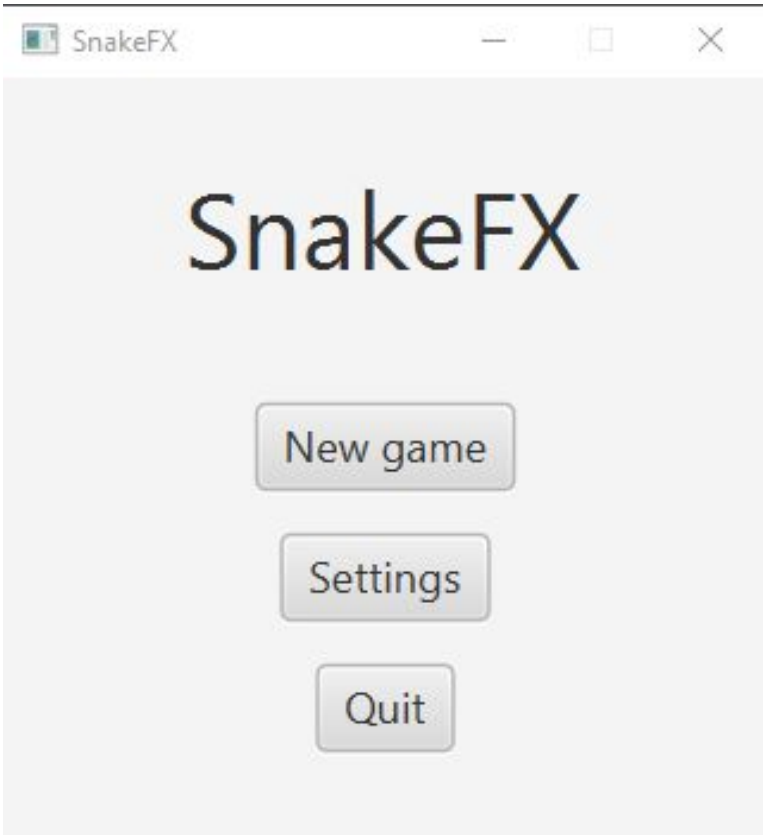


Figure 8. Snake Settings:

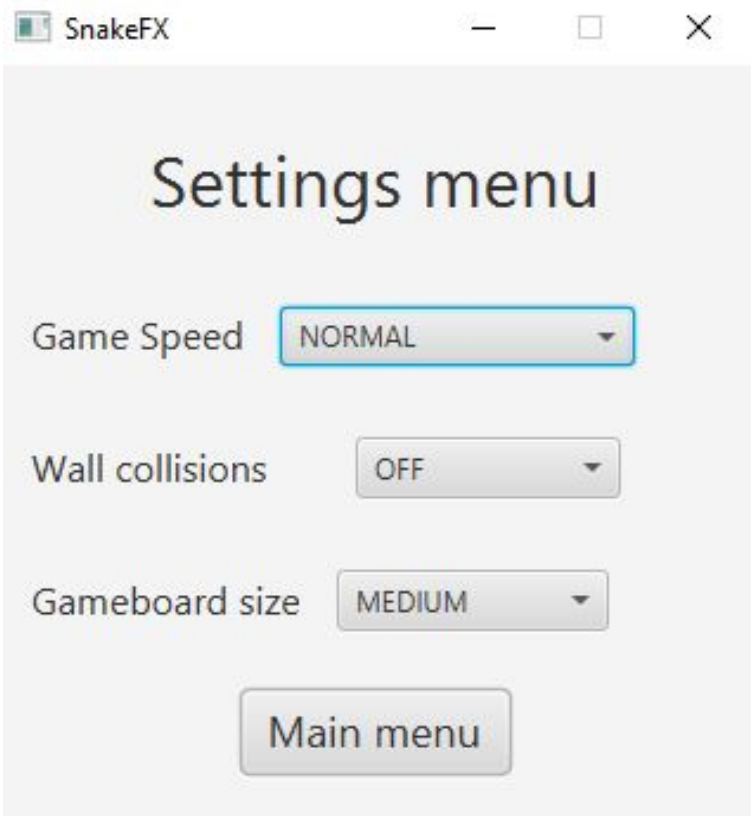


Figure 9. Snake Game:

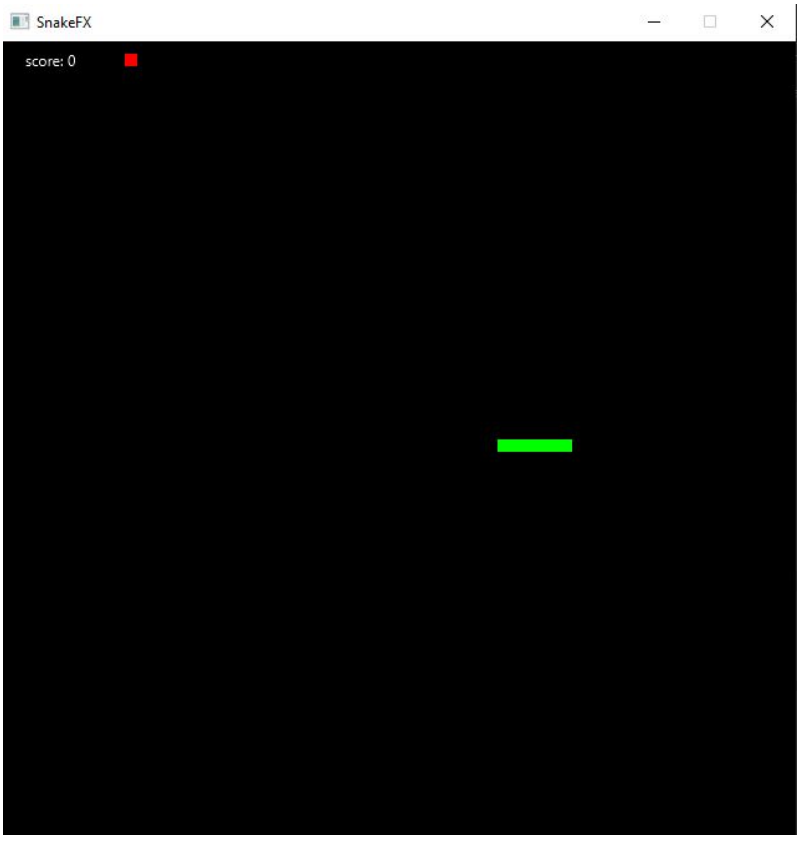


Figure 10. Space Invaders Game:

Figure 11. Space Invaders Pause:

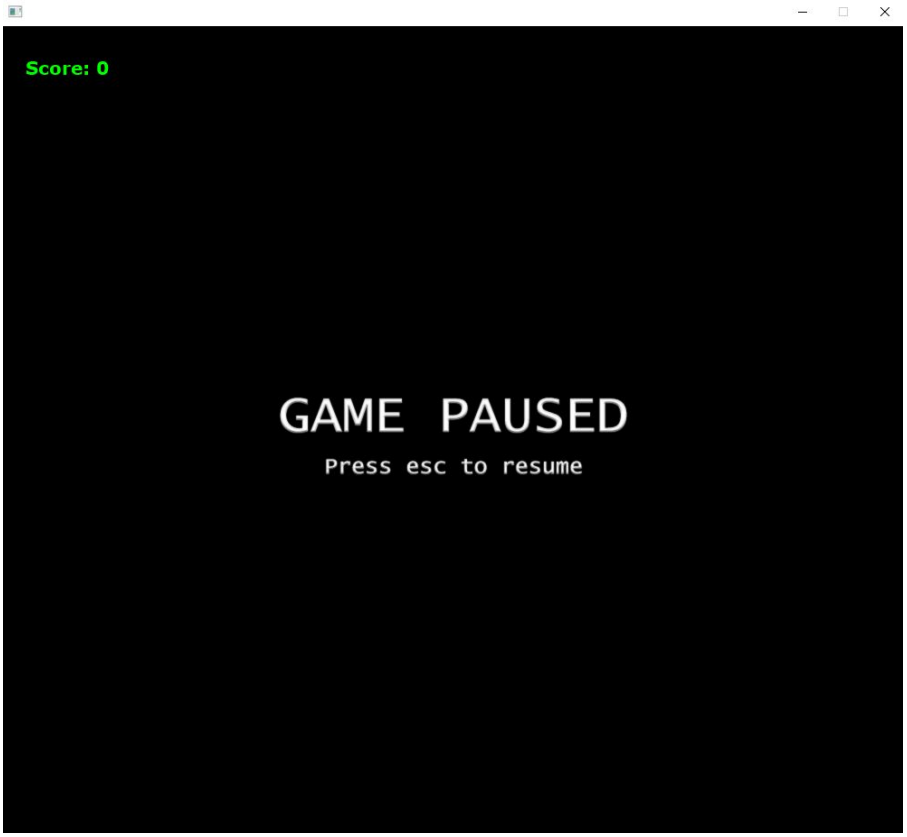
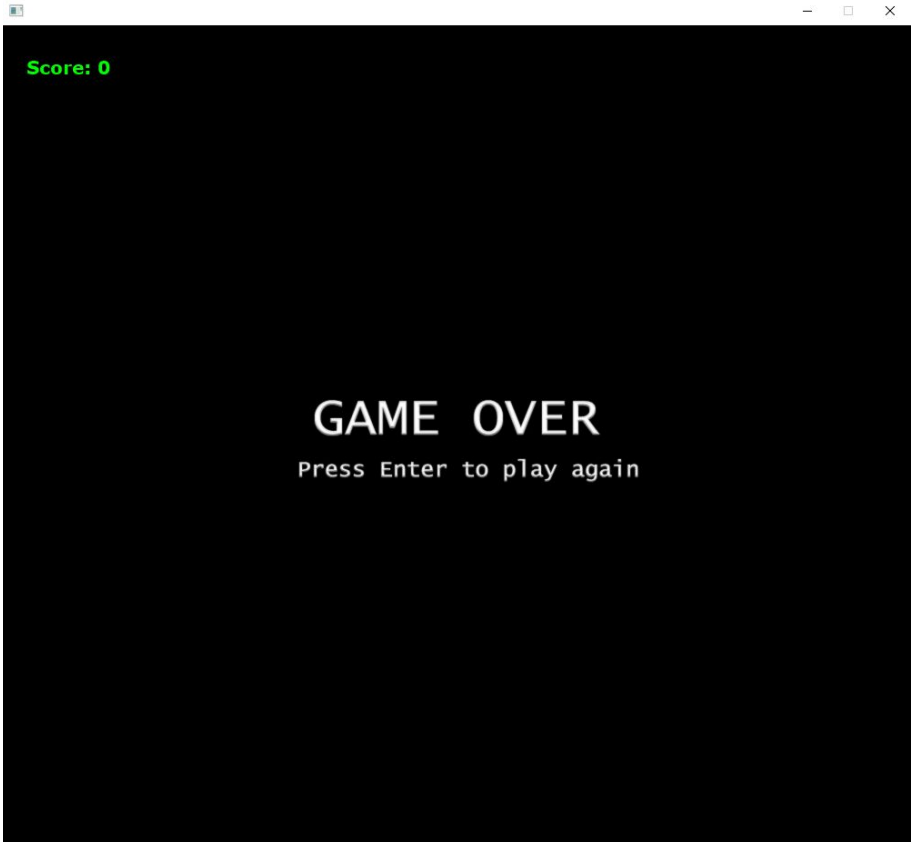


Figure 12. Space Invaders Game Over:



3. Domain model

3.1 Domain model

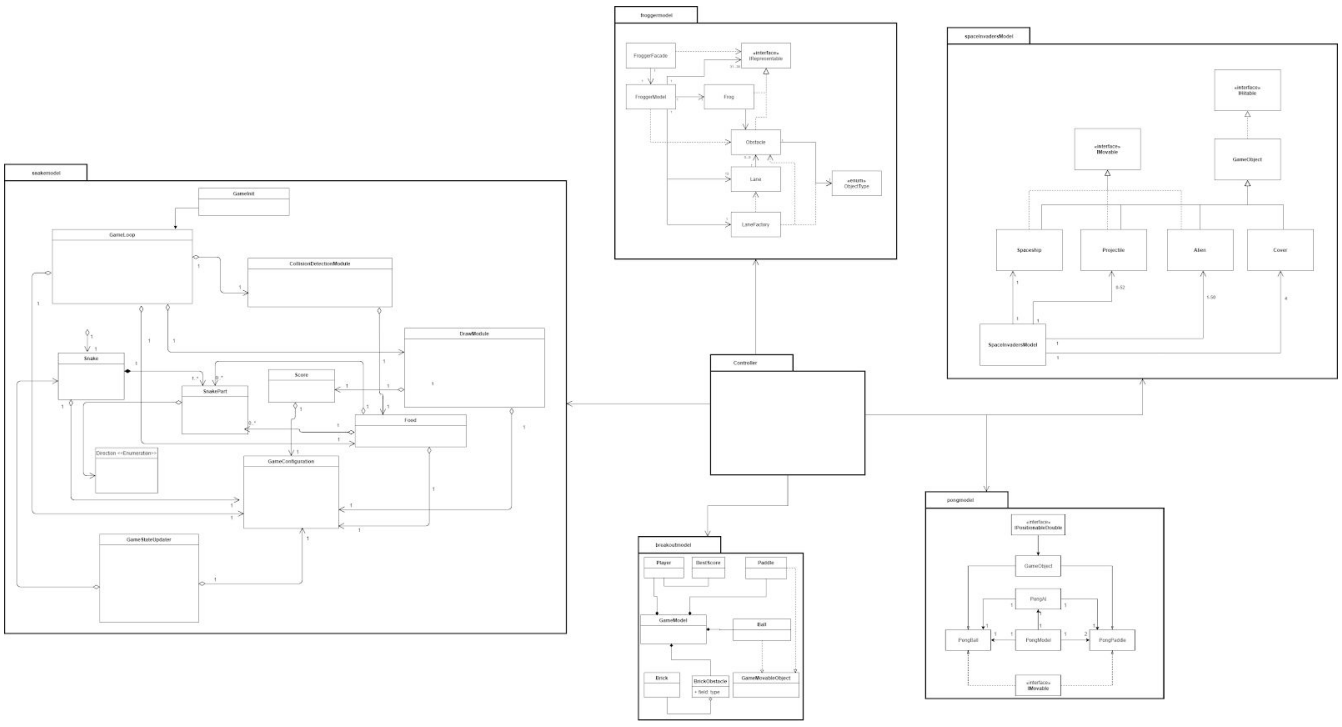


Figure 1. Domain Model[1]
(see attached pdf for more detailed view)

3.2 Class responsibilities

Explanation of responsibilities of classes in diagram.

Frogger:

- FroggerModel: Responsible for running the game.
- FroggerFacade: Responsible for representing the game of Frogger to other packages.
- Frog: Responsible for representing the player.
- Lane: Responsible for handling obstacles.
- Obstacle: Responsible for handling objects Frog needs to either avoid or in some way interact with.
- LaneFactory: Responsible for creating lanes for FroggerModel.
- ObstacleType: Responsible for representing the type of an Obstacle.

- IRepresentable: Responsible for handling objects that need visual representation.

Snake:

- CollisionDetectionModule: The class that detects a collision between the Snake, food and the wall.
- Direction: It represents the direction in which the player is moving.
- Food: Responsible for generating the food at random positions.
- GameStateUpdater: Responsible for the game state update.
- Score: Responsible to save and show the player's score.
- Snake: Responsible for representing the player.
- SnakePart: It represents the player parts.

Space Invaders

- SpaceinvadersModel: Model of the game, responsible for running and updating the game.
- Spaceship: Responsible for representing the player in the game.
- Alien class: Responsible for representing the enemies in the game.
- Projectile: Represents the projectiles fired by the player and the enemies
- GameObjectt: Abstract class representing an object in the game Alien, Projectile, Spaceship and Barrier all extend this class.
- IHitable: Interface which represents an object able to collide with other objects.
-
- IMovable: Interface which represents an object able to move.

Pong

- PongModel: Responsible for game logic and calling the update functions in PongBall and PongPaddle.
- PongBall: Responsible for representing the Pong Ball's position and moving the Ball in question.
- PongPaddle: Responsible for representing the Pong Paddles' position and moving the Paddle in question.
- PongAI: Responsible for making the right PongPaddle move up or down in response to the PongBall's movements.
- GameObject: Abstract class which has the properties for position and dimensions of both PongPaddle and PongBall.
- IMovable: Interface containing the Update function.
- IPositionable: Interface containing the getters of positional and dimensional variables.

Breakout

- GameModel : Class for Breakout controller. This class has some methods to detect collision between the ball and the wall, the ball and the paddle and the ball and bricks.
- Player: The class represents the player.

- bestScore: Responsible to save and show the player's score.
- Ball: The class has a move() method that moves the ball on the Board. If the ball hits the borders, the directions are changed accordingly.
- Paddle: This is the Paddle class. The paddle is controlled with left and right arrow keys. The paddle moves only in the horizontal direction, so we only update the x coordinate. If the ball misses the paddle and hits the bottom, we stop the game.
- Brick: This class creates object bricks for breakout. Yellow bricks earn one point each, green bricks earn two points, orange bricks earn three points and the top-level red bricks score four points each.
- BrickOnstacle: The class randomly creates and returns a brick obstacle

4 References

[1] Diagrams.net. [Online]. Available: https://app.diagrams.net/ Accessed: 2020-10-22.

5 Appendix

A pdf file named “DomainModel” can be found in the same folder as this file.