Marco Puig, Laura Waldron, Antonio Croissy, Hana Segura
October 21st, 2023
COP 4331

Application Requirements Edits

Use case are detailed and easy to understand.

Document lacks glossary and the platform. Add in.

Write down explicitly the platform and OS that you would be using. (Swing/Android..)

If you do a web application the UI still needs to be written in Java using the MVC architecture taught in class, i.e. Swing or Android. Client with web services that must also be written in Java.

Write the functional requirements in details.

Write a glossary with the important domain concepts, like transaction, product, seller, etc.

Revise the requirements document and make it the first section in Deliverable #2 - the design specification.

Option #2: Super Groovy Pong

Use Cases:

1 Player Controls Movement of the Paddle:
–the player moves their paddle to defend their goal
–the player presses arrow keys to move their paddle around ½ of the screen

2 Player Hits Puck:
–the player hits the puck with their paddle
–the paddle intercepts the puck to send it in the right direction
–the puck's speed increases with each successful hit
–a basic line render controls the paddle speed
–we start with a cold color and increment to hotter colors
–AI responds to the puck by hitting it back to also defend its goal.

3 Pause/Resume:
–the player wants to pause the game
–pressing the "Pause" button, or the P key will temporarily halt the game
–press the same button to continue the game

4 Scoring:
–the player scores one point per round
–when the puck passes the opponent's paddle & enters the opponent's goal, they score a point.
–score is updated on the screen

5 Game Over:
–the game ends when the player reaches best of 5 (first to 3).
–the "Game Over" screen is displayed (Player or AI wins).
–players can return to the main menu, or restart the game

6 Game Restart:
–the player clicks "Restart" to begin a new game, for convenience.
–after the game ends, they can click the "Restart" button or use the key (R key) to initiate a new game

7 Game Exit:
–the player decides to exit the game
–the player can click the "Exit Game" button or use the (Esc) key to close the game.
–the game closes and players are returned to their desktop screen

8 Game Instructions:
–the player must understand how to play
–the player can access an "Instructions" screen providing information on controlling paddles, scoring points, or special game rules

Platform:

For platform, we will be developing our application using LibGDX. LibGDX is a library for Java, written in Java, to aid in Game Development. It comes with a sprite rendering engine, window handling, and additionally built in libraries that will help us develop our game/project.

OS:

For Operating System, since we are developing our project using LibGDX, our project will be targeted for the Windows Operating System.

Glossary

Player: The human using the computer and controlling the motions of the paddle in the game.
Opponent Player: Either an AI player or another human player, depending on the choice, who competes with the player to score as many Goals as possible.
Paddle: The item the player uses to hit the puck away from their goal.
Puck: A object the player will hit back and forth with their paddle
Goal: The place the player is trying to put the puck. When the goal reaches 4 points, the respective player wins.

Restart: The button that resets the game to its original state
Pause: The button that pauses the game, or resumes the game if already paused

Design Specification

Submit a pdf document on the Pages project page with the following:

Updated Application Requirements. Incorporate the analysis comments (#1) from the instructor.
CRC cards:
identify classes for this application
assign responsibilities
find collaborators

UML diagrams:
class diagrams that show class attributes, methods and relationships
sequence diagrams, for interactions between objects, according to use case scenarios. Identify the Use Case scenario for each sequence diagram.
State diagrams for classes that have non-trivial behavior

Deadline on 10/28, 11:59PM. Upload just the PDF file on the Pages wiki, on the group Canvas page and on the group's GitHub page.

Use Case 1 Player controls movement of the paddle–Laura Waldron

CRC Cards:
Actor: Player (human)

Responsibilities:
- Control the movement of the paddle

Collaborators:
- Class Paddle

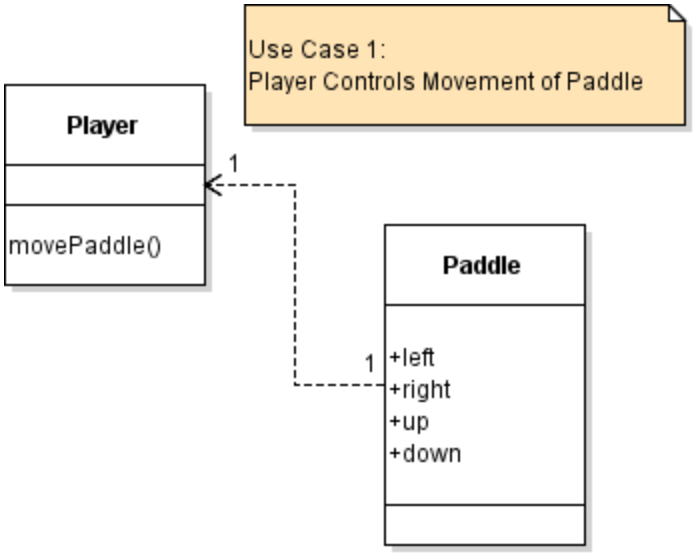
Methods:
- movePaddle()

Class Paddle

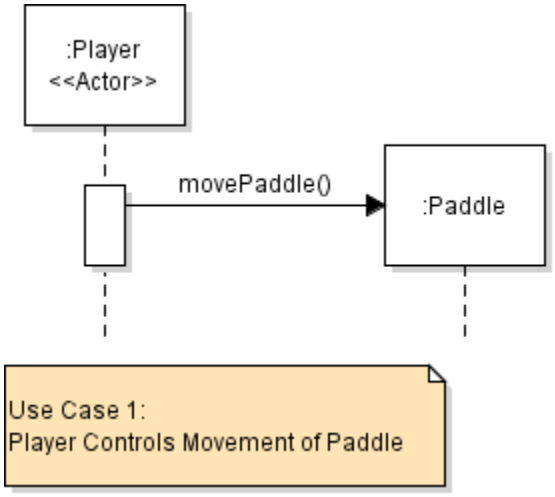
Responsibilities:
- Move the puck from one location to another

Collaborators:
- Class Puck

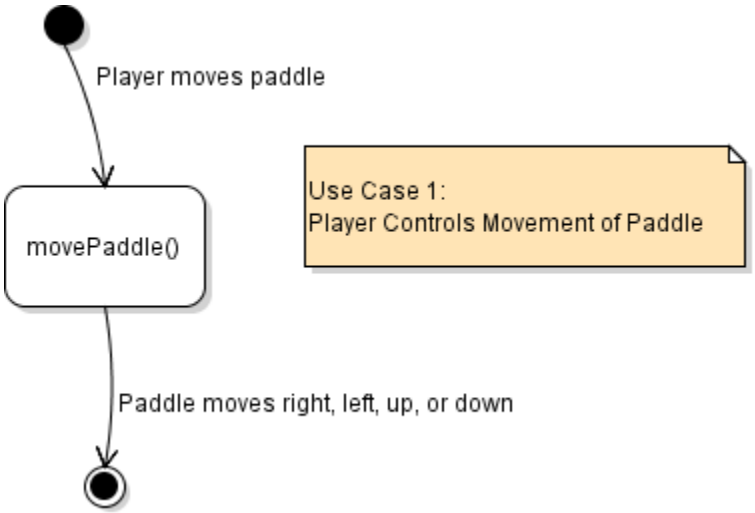
Class Diagram:



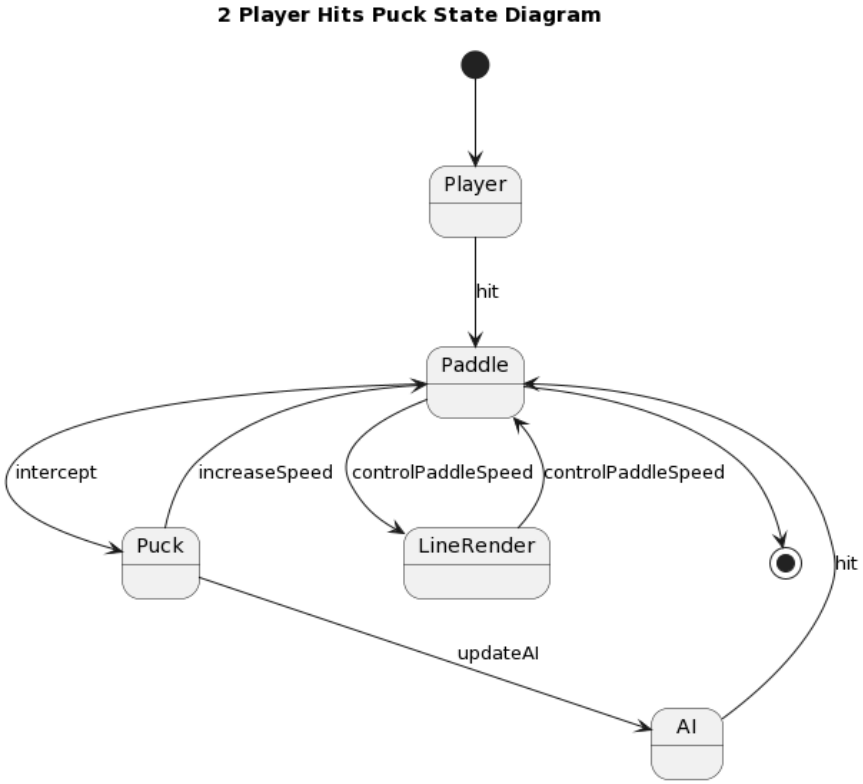
Sequence Diagram:

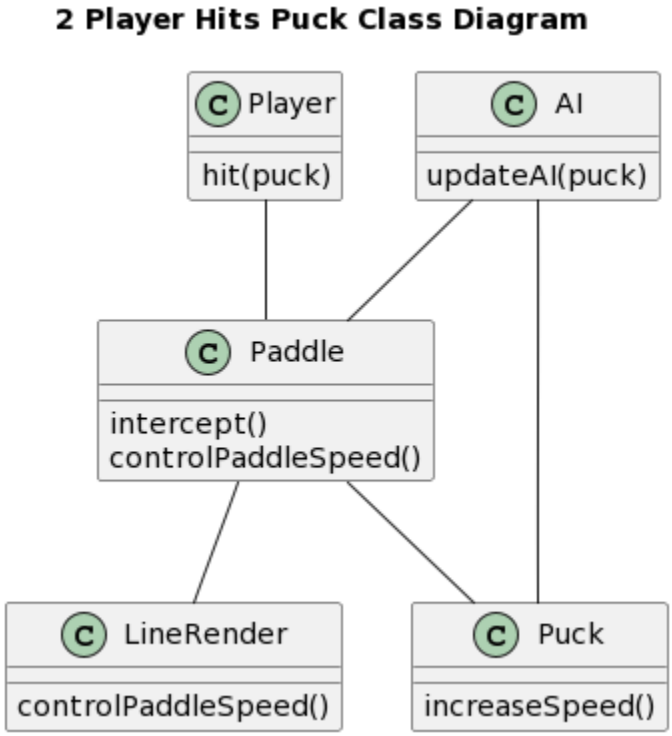


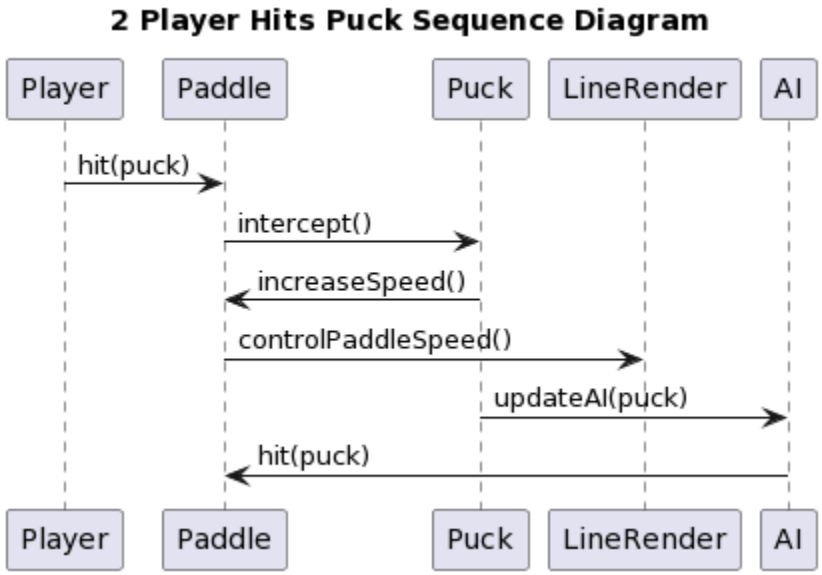
State Diagram:



Use Case 2 Player hits puck - Marco Puig







Use Case: Player Hits Pucks
Class Name: Player

Responsibilities:
- HitPuck()
- IncreasePuckSpeed()

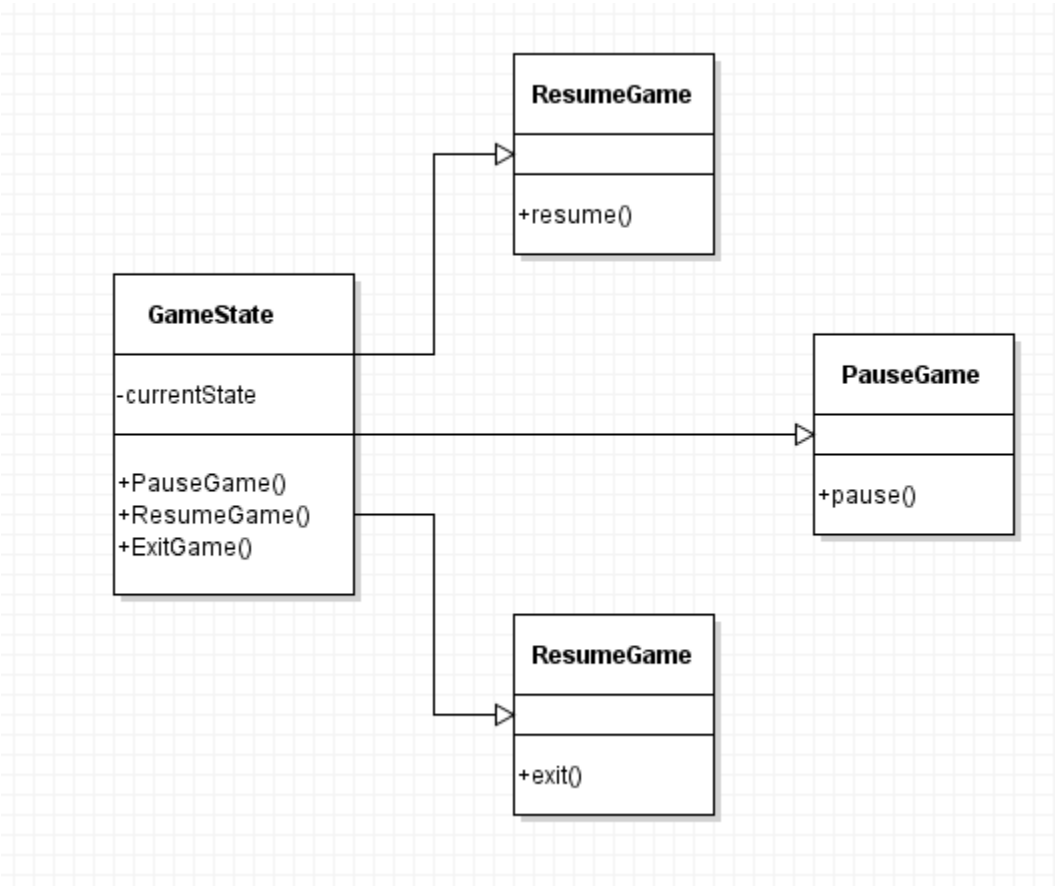
Collaborators:
- Puck

<u>Use Case 3 Pause/Resume – Hana Segura</u>
CRC cards

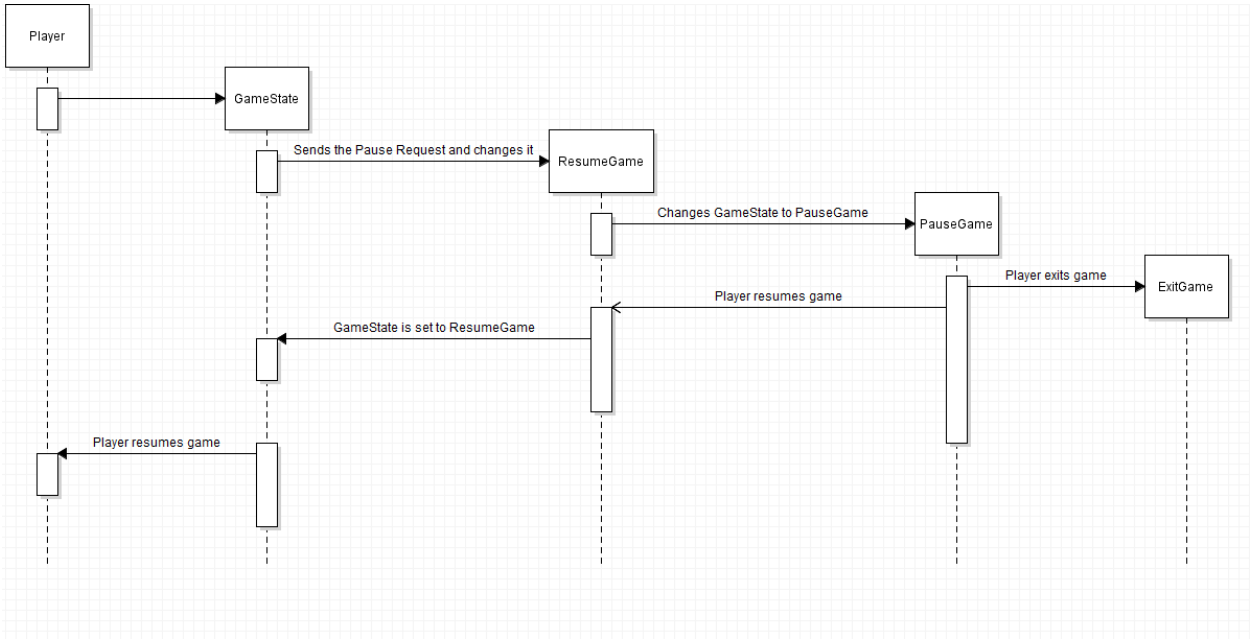
- Class: GameState
  - Responsibilities:
    - Maintain the current state of the game.
    - Allow changes to the game state.
  - Collaborators:
    - PauseGame
    - ResumeGame
    - ExitGame
- Class: ResumeGame
  - Responsibilities:
    - Handle the logic for resuming the game.
  - Collaborators:

        - GameState
- Class: PauseGame
    - Responsibilities:
        - Handle the logic for pausing the game.
    - Collaborators:
        - GameState
- Class: ExitGame
    - Responsibilities:
        - Handle the logic for exiting the game.
    - Collaborators:
        - GameState

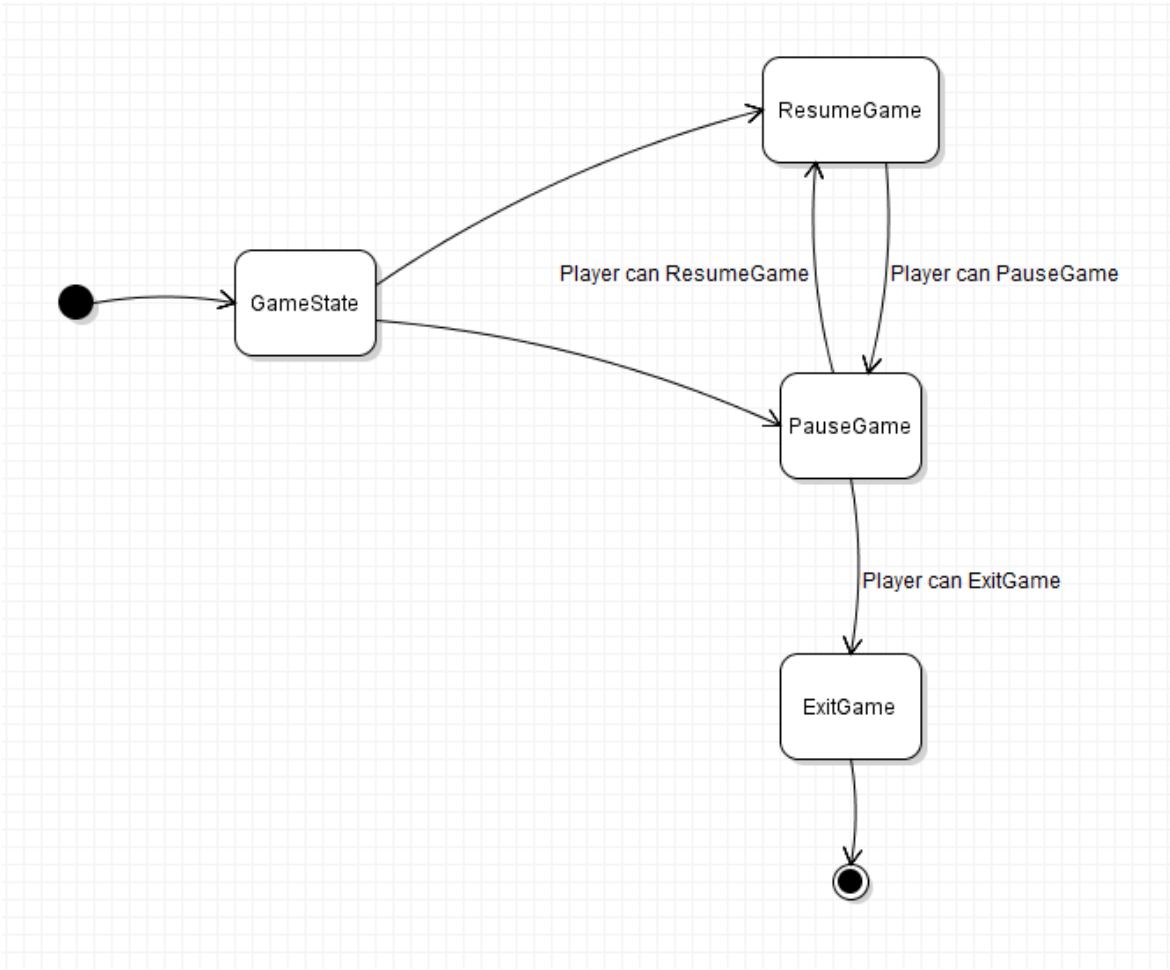
Class Diagram:



Sequence Diagram:



State Diagram:



Use Case 4 Scoring –Laura Waldron

CRC cards

Actor: Player (human)

Responsibilities:

- Score one point (max) per round

Collaborators:

- Class Puck

Methods:

- hitPuck()

Class Puck

Responsibilities:

- Enter the opponent's goal to score a point

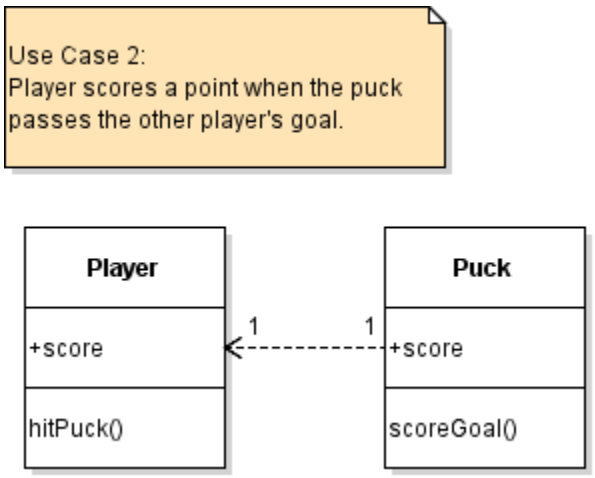
Collaborators:

- Class Player

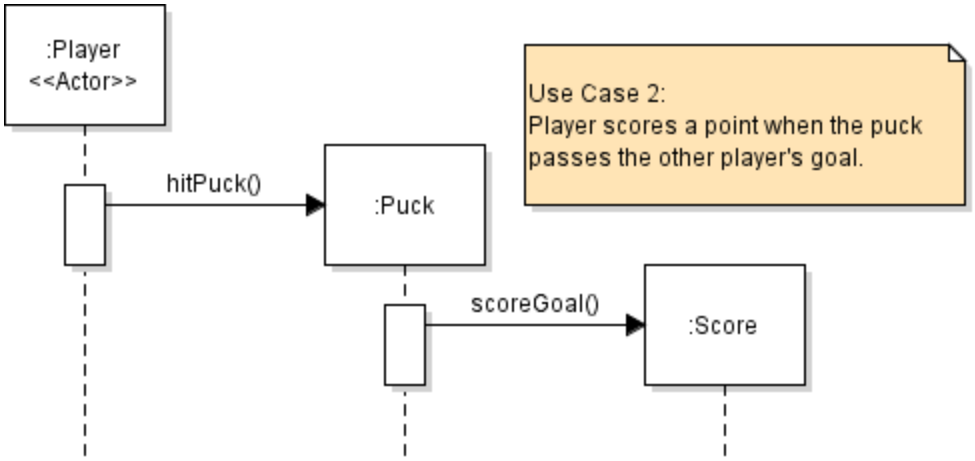
Methods:

- scoreGoal()

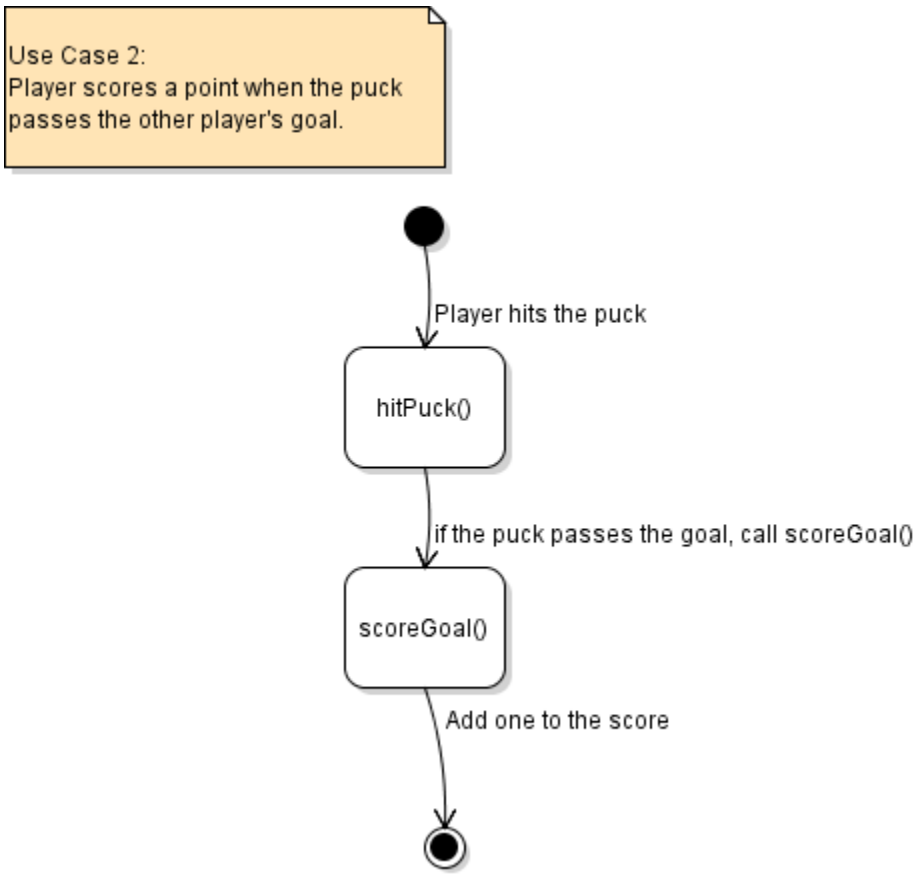
Class Diagram:



Sequence Diagram:



State Diagram:



<u>Use Case 5 Game Over - Antonio Croissy</u>

CRC Card
gameOver
responsibilities:

- display game over screen

Collaborators:

- game

Actor: Player (human)

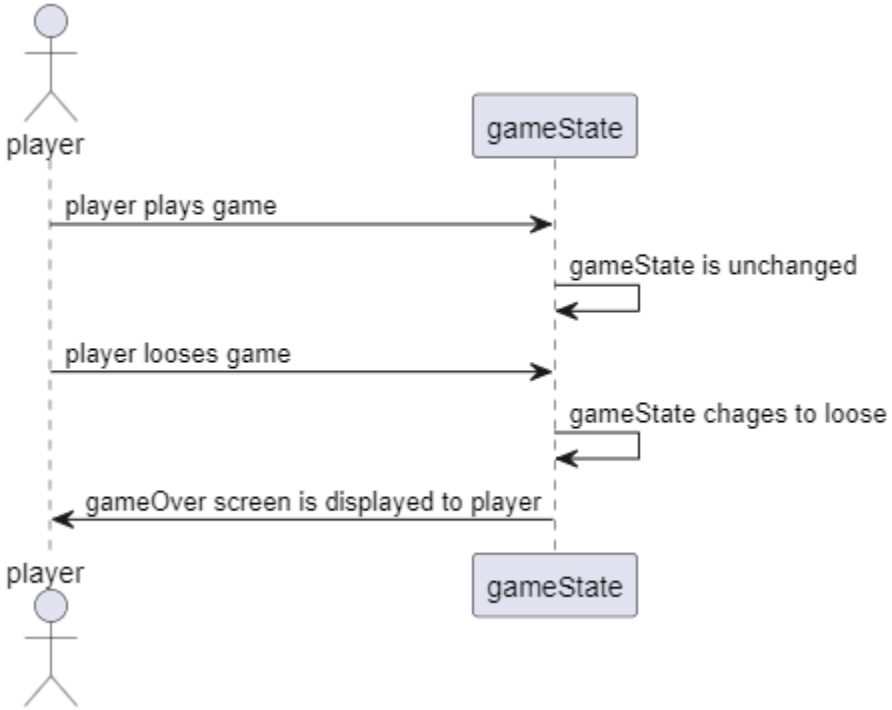
responsibilities:

- plays game

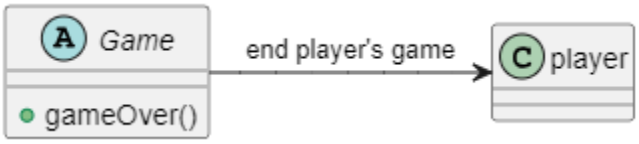
Collaborators:

- game
- game interface

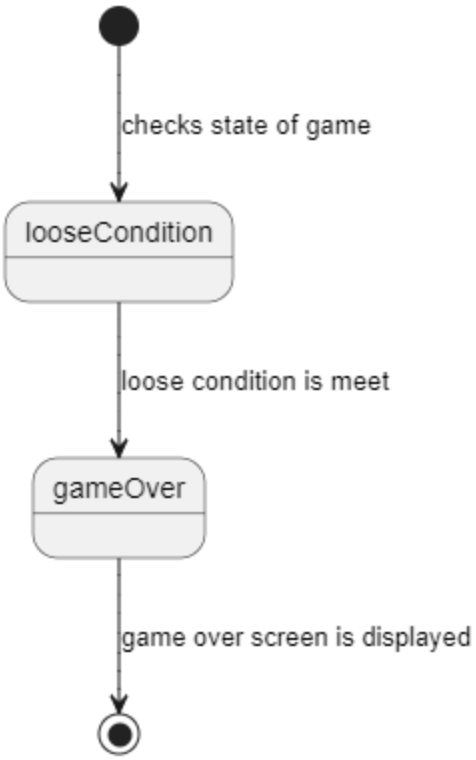
Sequence



Class



State



Use Case 6 Game Restart - Antonio Croissy

CRC Card
gameRestart
responsibilities:
- reset game back to initial conditions

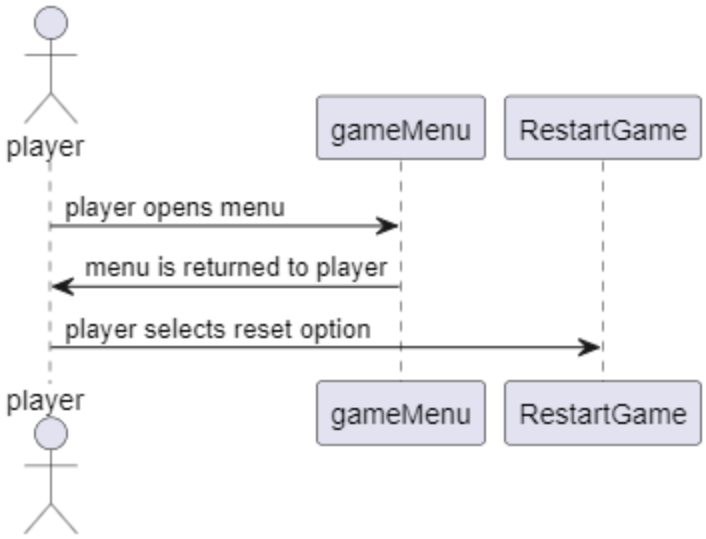
Collaborators:
- game menu

Methods:
- Restart()

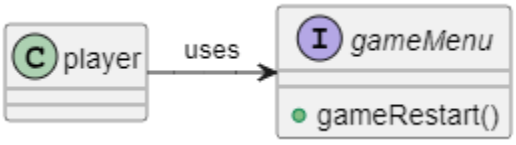
Actor: Player (human)
responsibilities:
- pick option from game menu

Collaborators:
- game interface

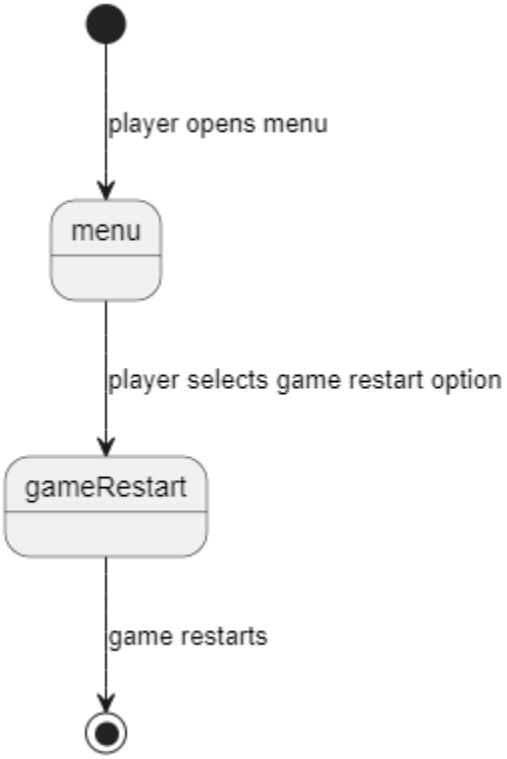
Sequence



Class



State

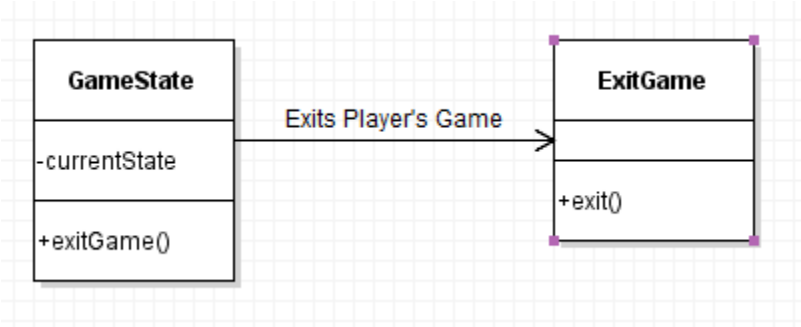


<u>Use Case 7 Game Exit – Hana Segura</u>

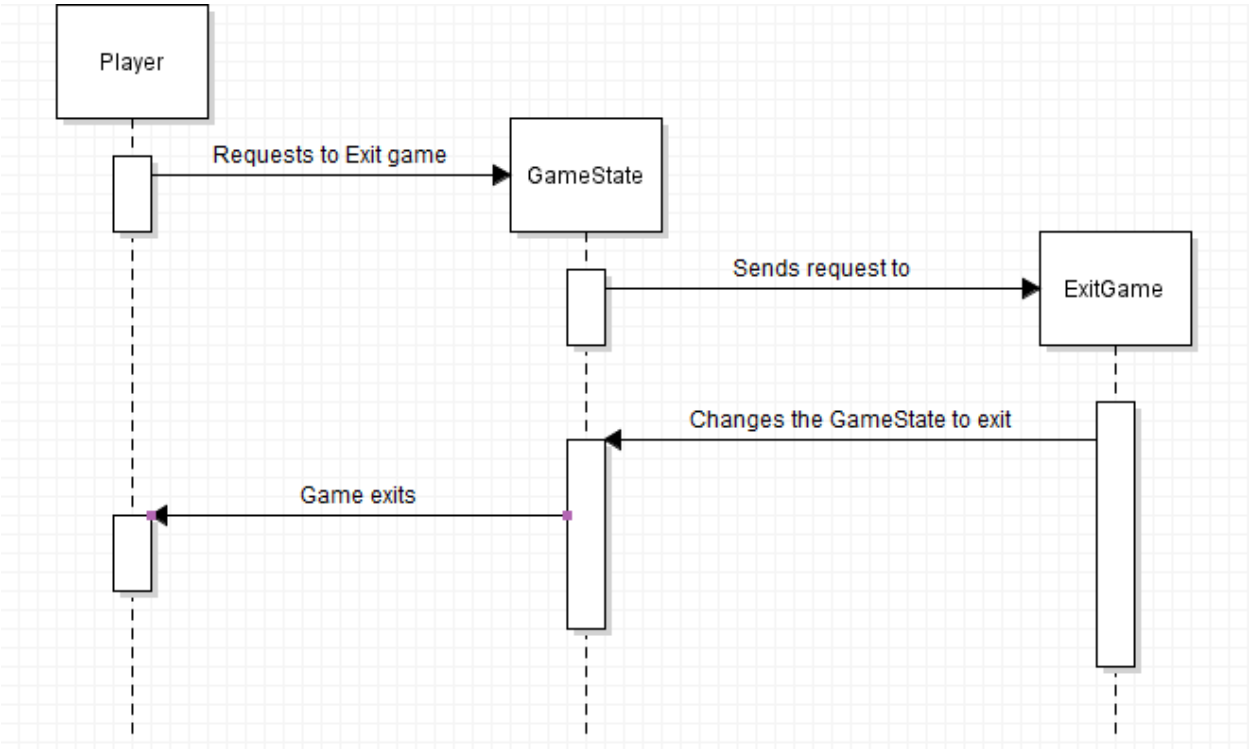
CRC cards

- Class: GameState
  - Responsibilities:
    - Maintain the current state of the game.
    - Allow changes to the game state.
  - Collaborators:
    - ExitGame
- Class: ExitGame
  - Responsibilities:
    - Handle the logic for exiting the game.
  - Collaborators:
    - GameState

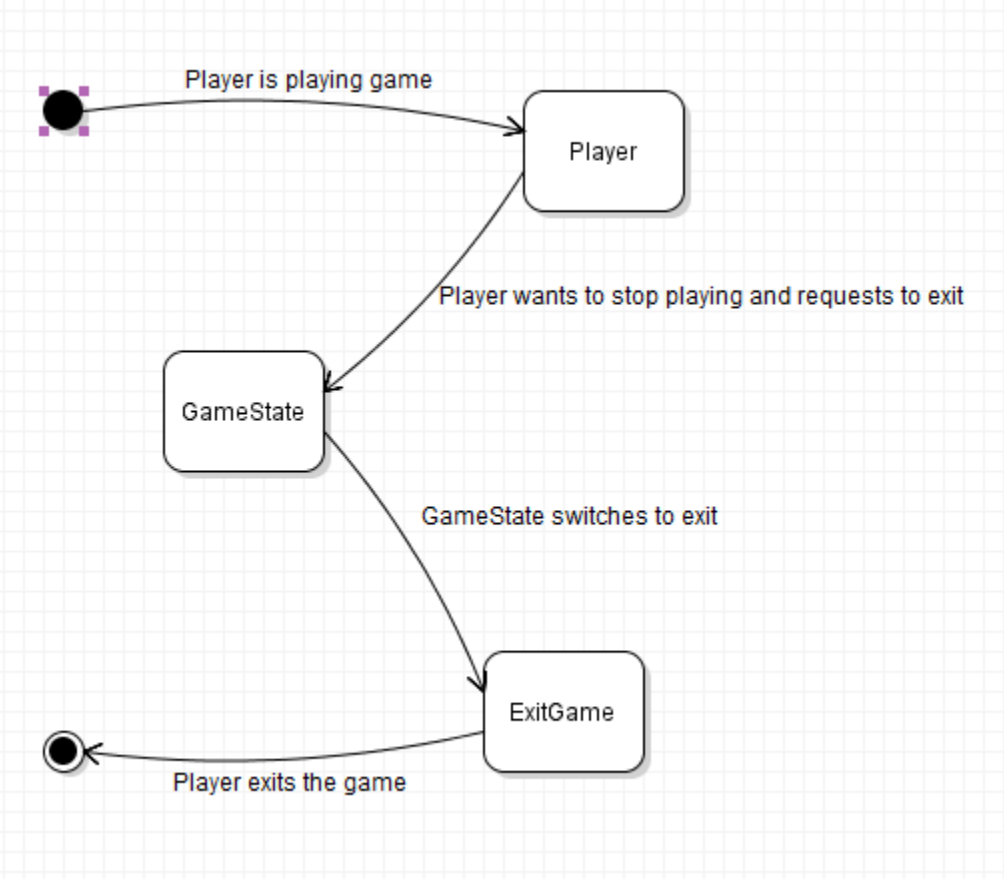
Class Diagram:



Sequence Diagram:

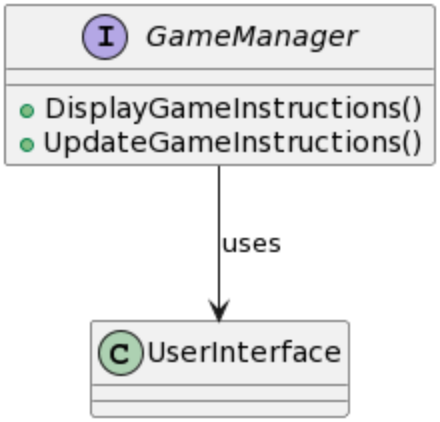


State Diagram:

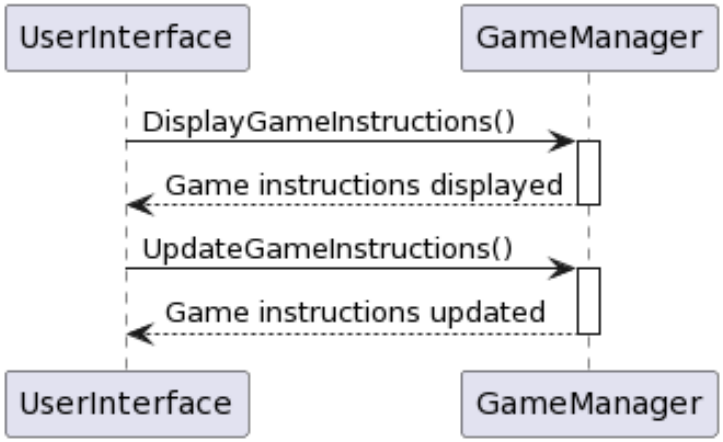


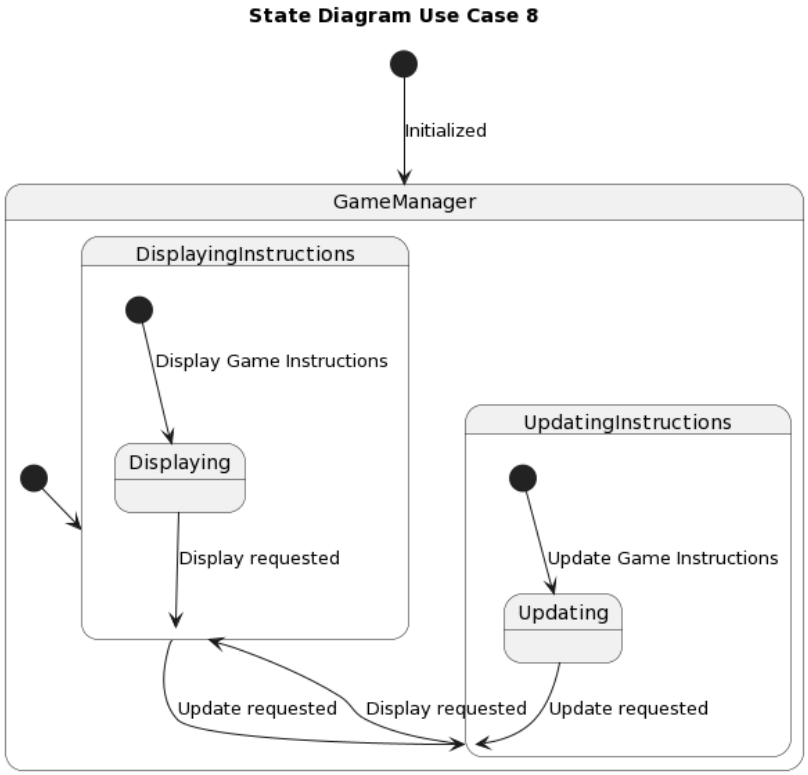
<u>Use Case 8 Game Instructions - Marco Puig</u>
Class Diagram

Use Case 8 Game Instructions



Sequence Diagram





Use Case: Game Instructions
Class Name: GameManager

Responsibilities:
- DisplayGameInstructions()
- UpdateGameInstructions()

Collaborators:
- UserInterface