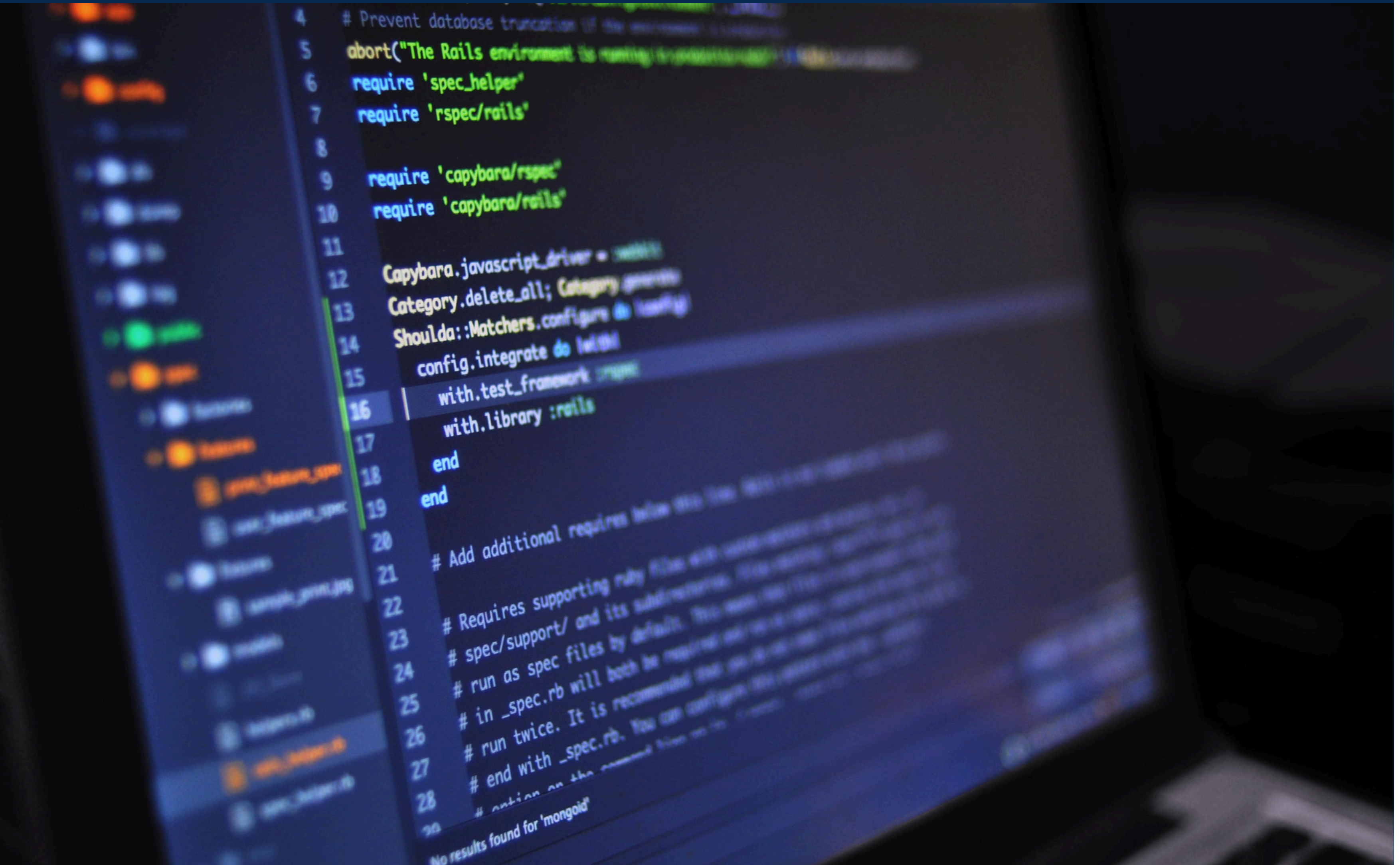
SPACE INVADERS

MARIO ÁLVAREZ GRACIA
SIMON BOSQUE GONI

PYTHON PROGRAMMING

SUMMARY



```
# Prevent database truncation [illegible]
abort("The Rails environment [illegible]
require 'spec_helper'
require 'rspec/rails'

require 'capybara/rspec'
require 'capybara/rails'

Capybara.javascript_driver = [illegible]
Category.delete_all; [illegible]
Shoulda::Matchers.configure [illegible]
  config.integrate do [illegible]
    with.test_framework [illegible]
    with.library :rails
  end
end

# Add additional requires [illegible]

# Requires supporting [illegible]
# spec/support/ and its [illegible]
# run as spec files by [illegible]
# in _spec.rb will both be [illegible]
# run twice. It is [illegible]
# end with _spec.rb. [illegible]
```

No results found for 'mongoid'

- INTRODUCTION
- THEORICAL PART
- PRACTICAL PART
- CONCLUSION

1. INTRODUCTION

1. INTRODUCTION

This project involves creating a simple 2D game using Python where a green spaceship controlled by the player must shoot and destroy invading aliens. The game utilizes the Pygame library to handle graphics and user input, offering an engaging experience reminiscent of classic arcade games.

PYTHON PROGRAMMING

1. INTRODUCTION

Aim

The primary aim of this project is to develop a basic interactive game where the player controls a spaceship to defend against waves of alien invaders. The focus is on implementing game mechanics, including movement, shooting, collision detection, and scoring.

Scope

The scope of this project includes:

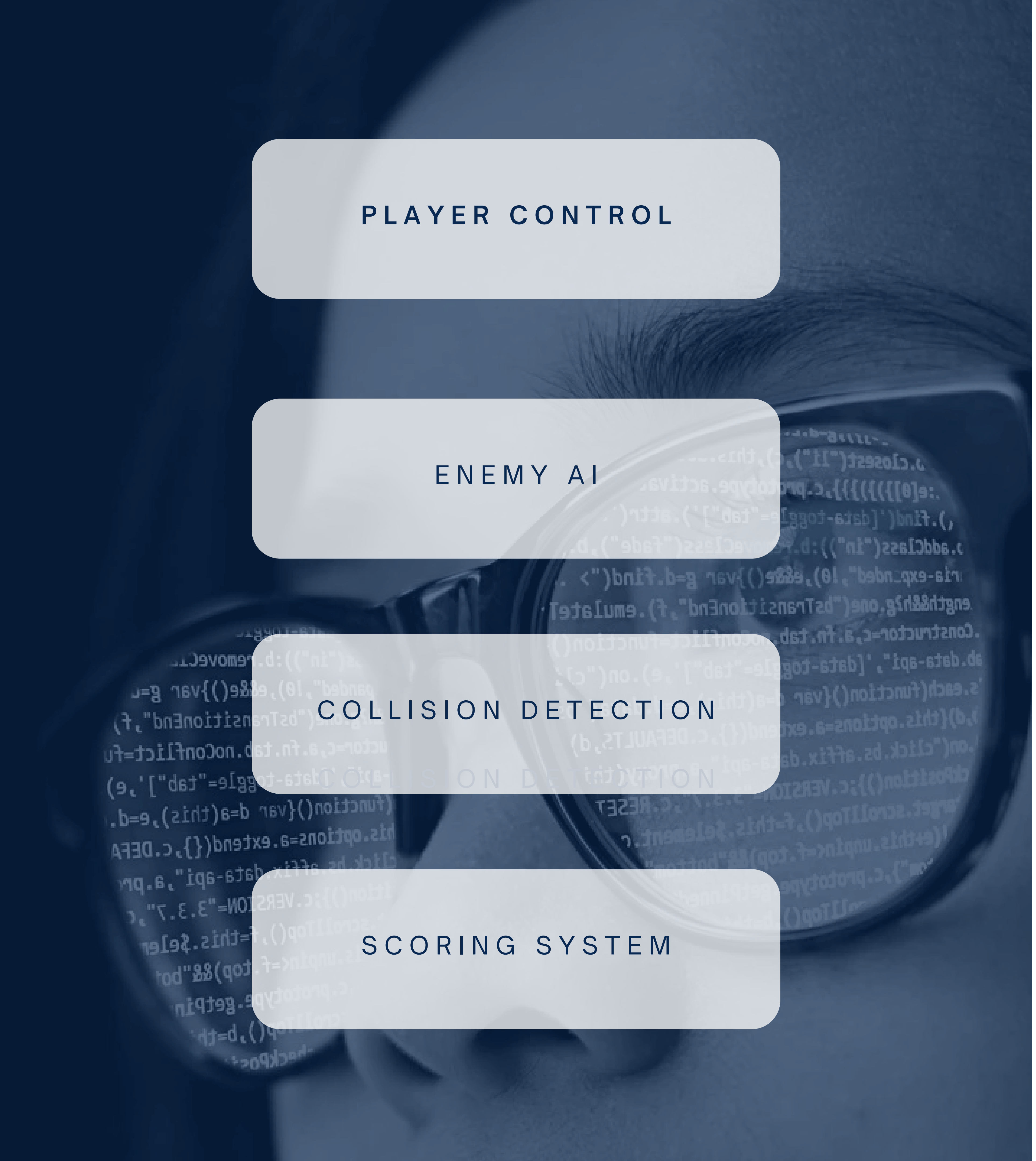
- Developing a player-controlled spaceship that can move left, right, and shoot.
- Creating alien invaders that move towards the player and must be destroyed.
- Implementing basic game mechanics such as collision detection and scoring.
- Using the Pygame library for rendering graphics and handling user input.

Methodology

The project is developed using Python, leveraging the Pygame library for game development. The process involved:

- Designing the game layout and mechanics.
- Coding the spaceship and alien behaviors.
- Implementing collision detection and scoring systems.
- Testing and debugging to ensure smooth gameplay.

2. THEORICAL PART



PLAYER CONTROL

ENEMY AI

COLLISION DETECTION

SCORING SYSTEM

2. THEORICAL PART

Game Design Theory

Game design involves creating interactive experiences that engage players. Key elements include:

- **Player Control:** Ensuring responsive and intuitive controls for the spaceship.

- **Enemy AI:** Designing predictable yet challenging patterns for alien movement.

- **Collision Detection**: Implementing algorithms to detect interactions between the spaceship’s bullets and the aliens.

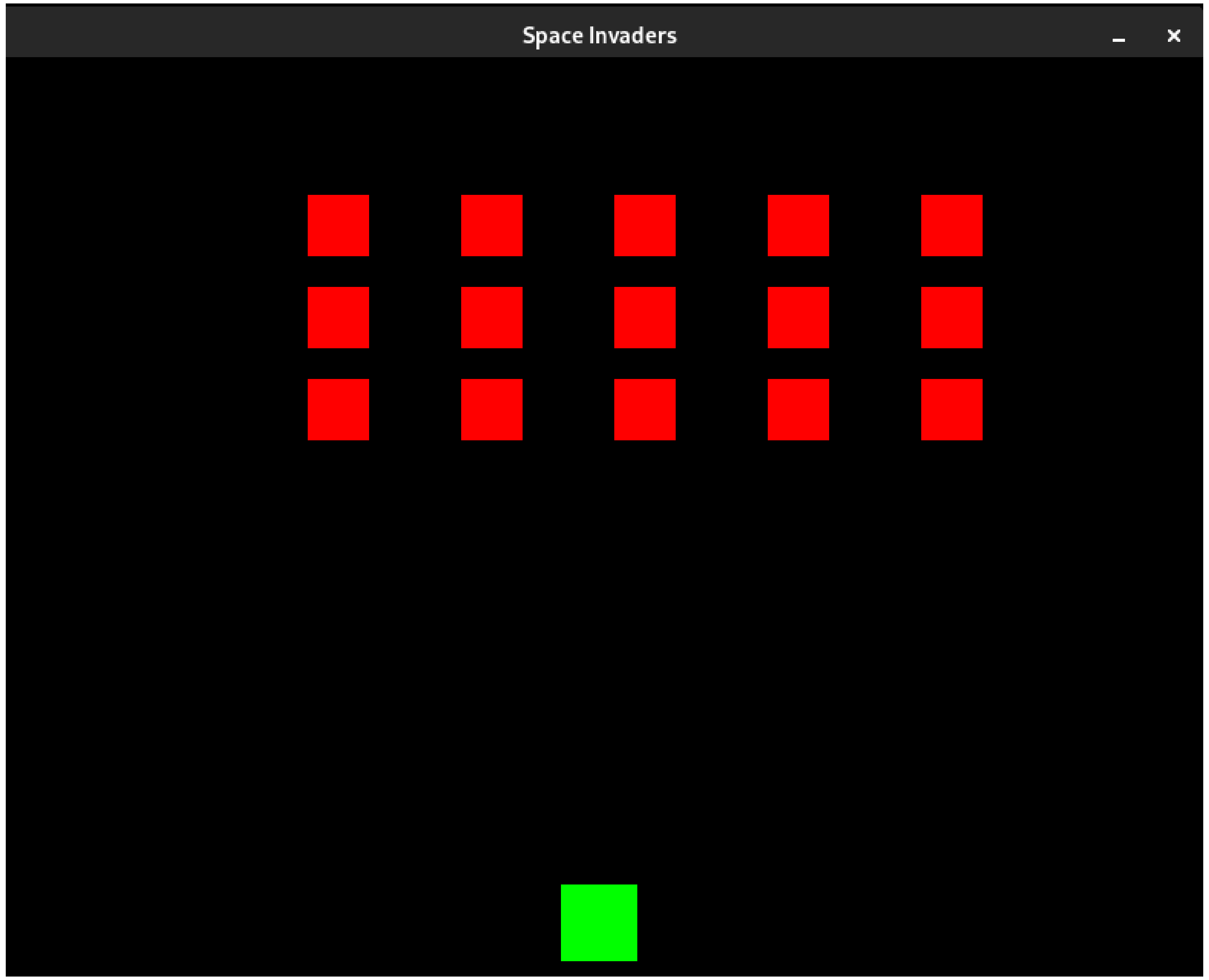
- **Scoring System:** Rewarding players for destroying aliens and achieving high scores.

3. PRACTICAL PART

3. PRACTICAL PART

Environment

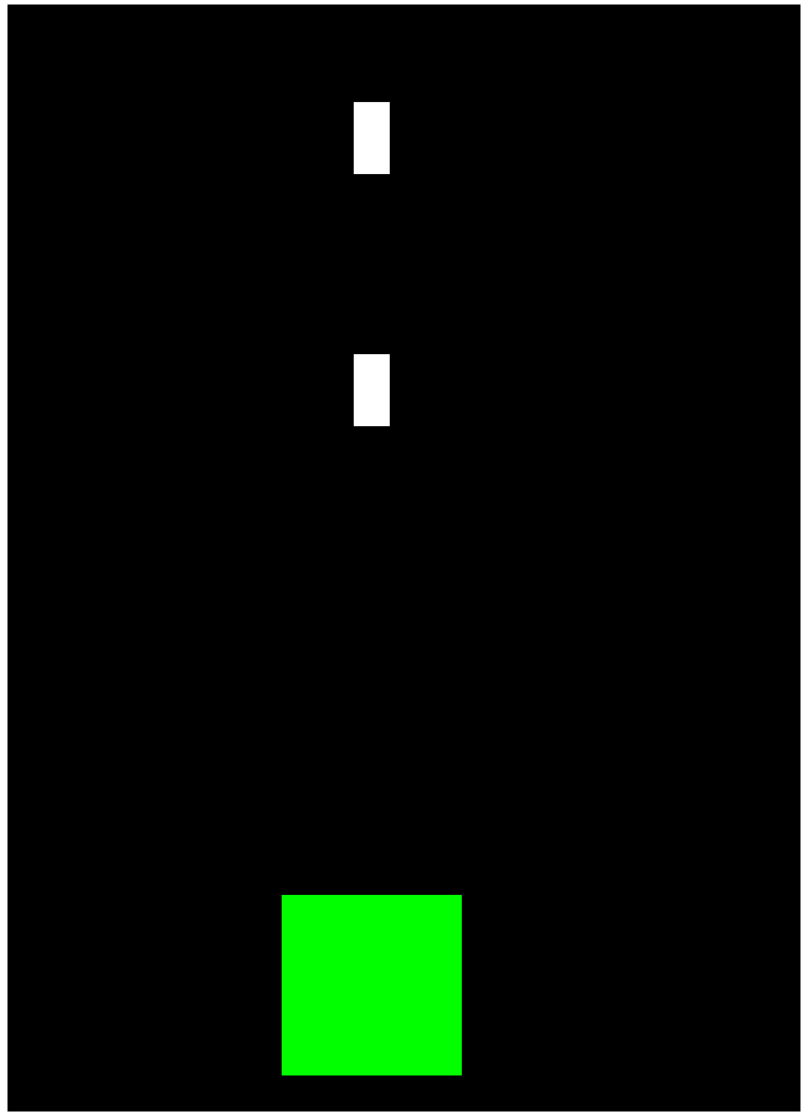
We set up a Pygame window with a background image representing space. The spaceship and alien images are loaded and rendered on the screen. The spaceship is controlled using keyboard inputs, allowing it to move left, right, and shoot bullets.



3. PRACTICAL PART

Player Control

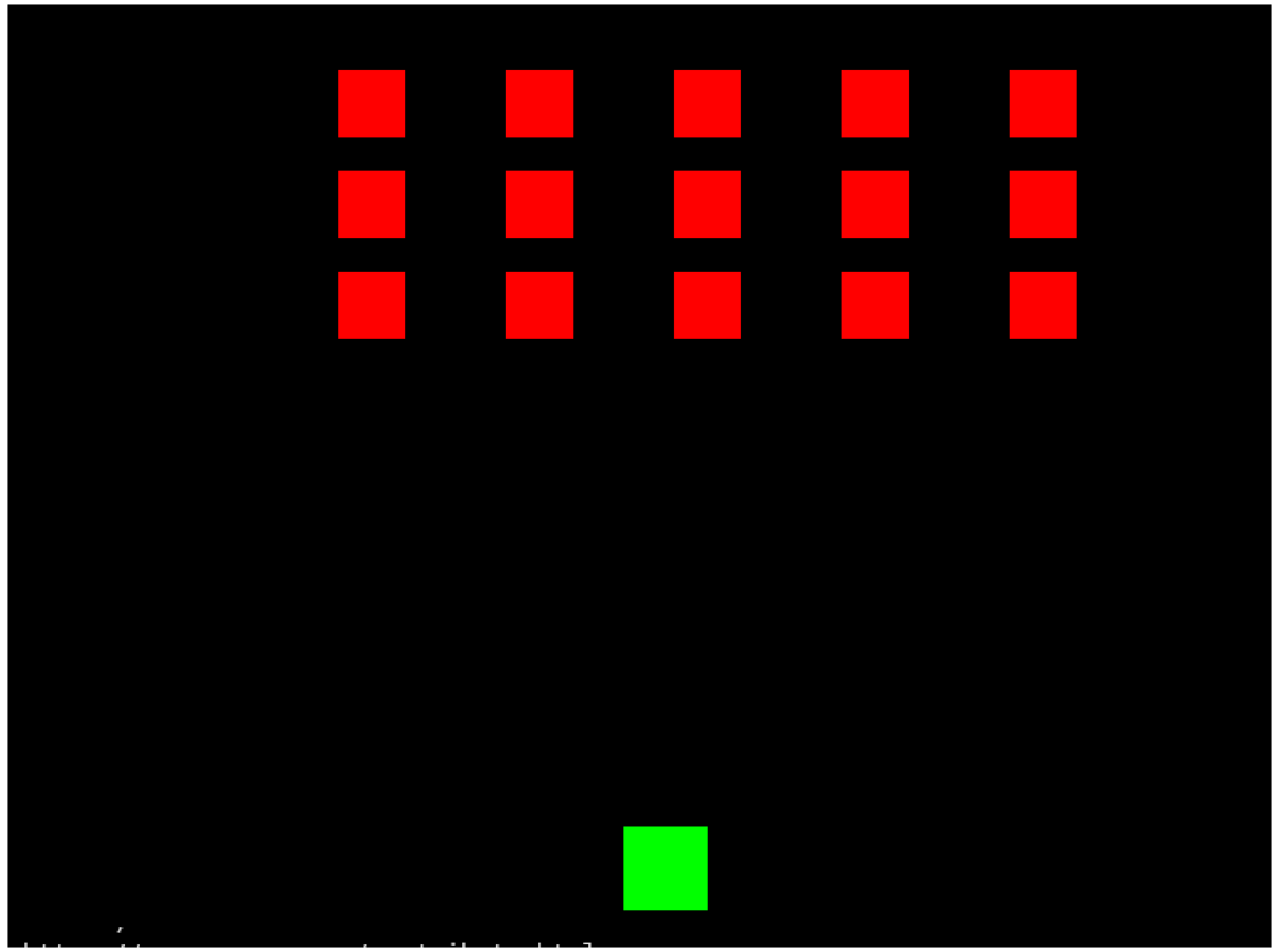
The spaceship’s movement is controlled by capturing keyboard events. The left and right arrow keys move the spaceship horizontally, while the space bar triggers shooting.

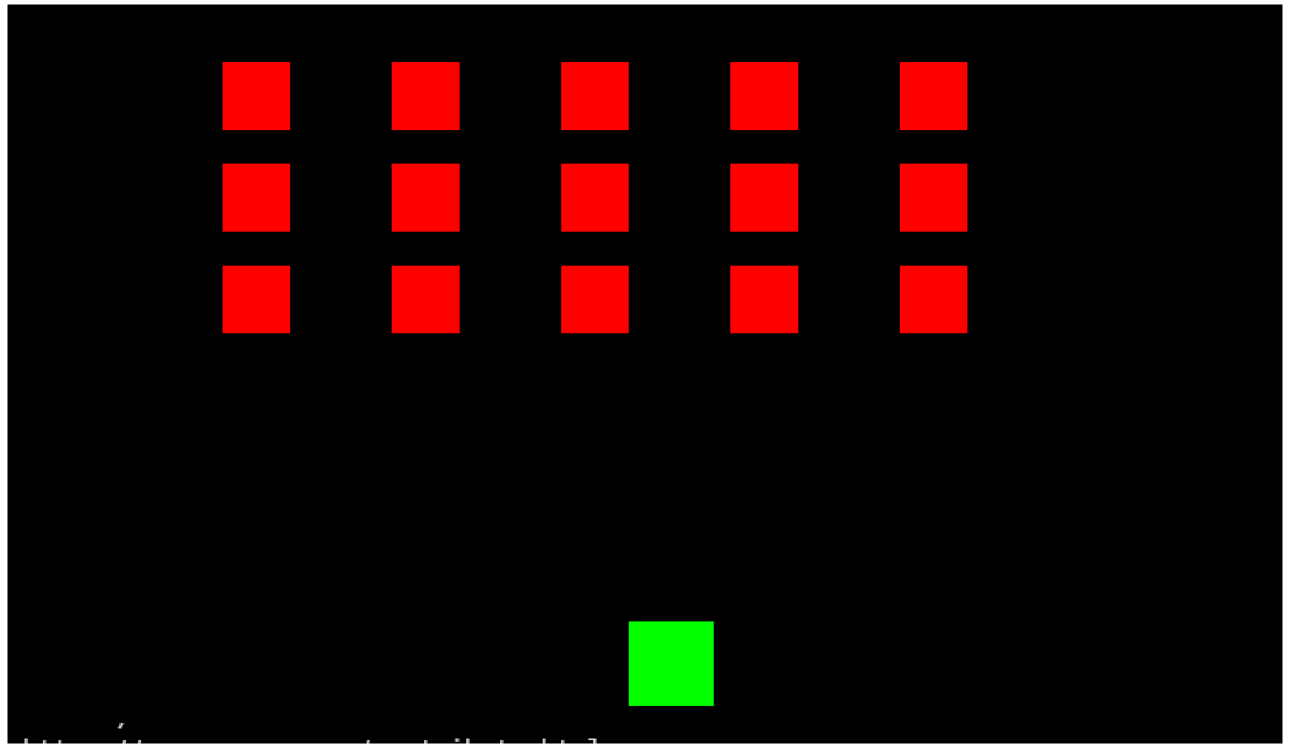


3. PRACTICAL PART

Alien Behavior

Aliens are programmed to move horizontally across the screen and descend towards the player at intervals. The position of the aliens is updated in each frame, and they reset upon reaching the screen edges.

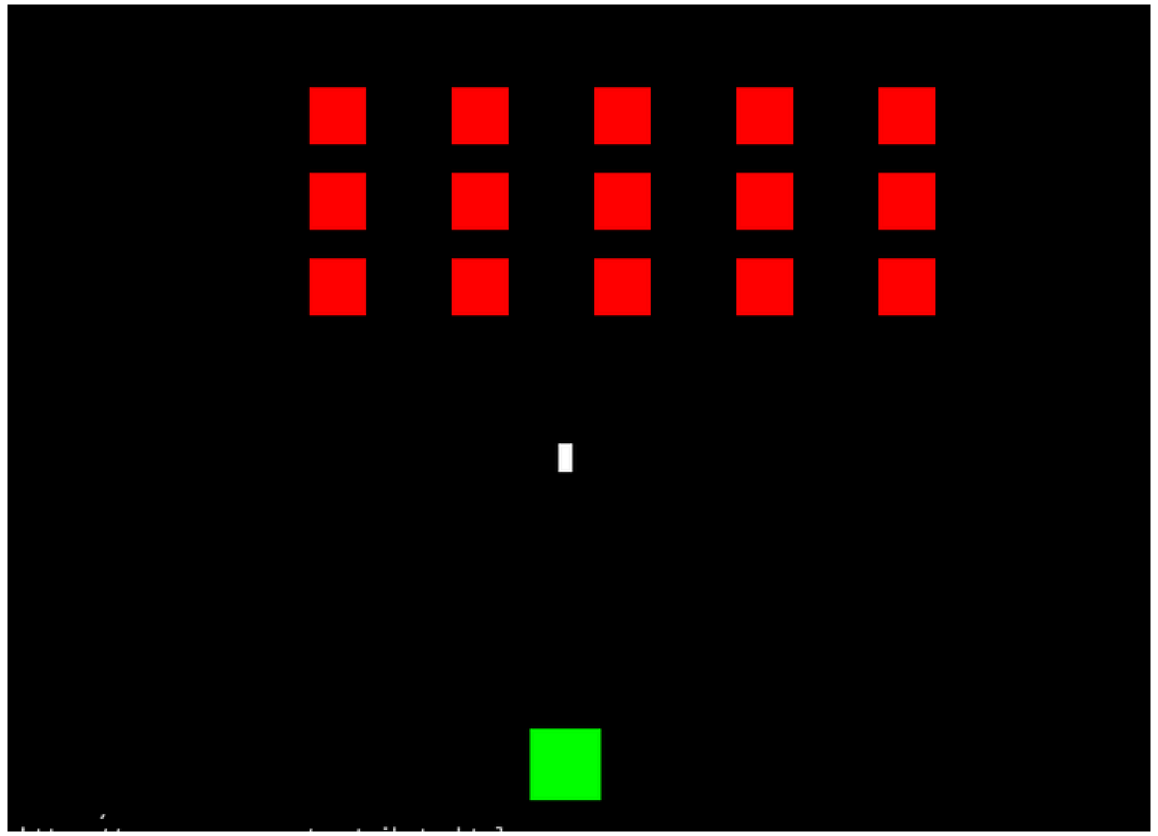


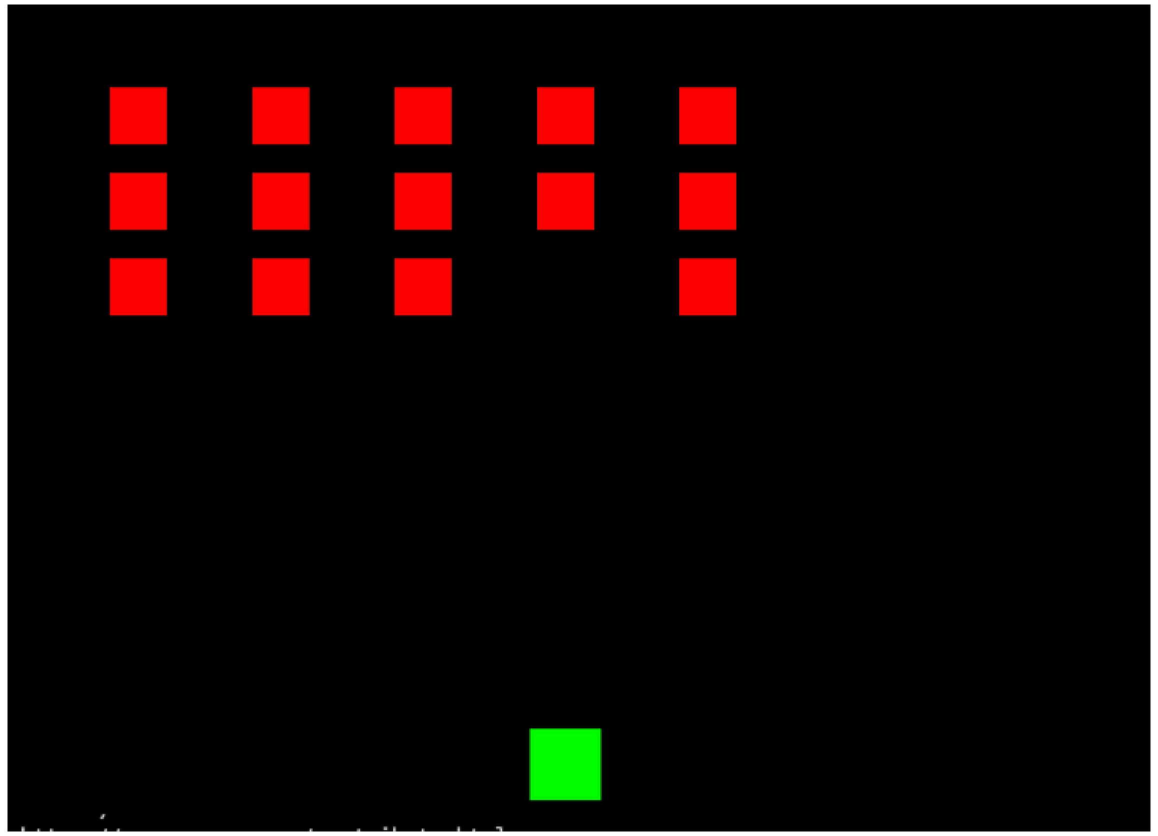


3. PRACTICAL PART

Collision Detection

Collision detection is implemented to determine when a bullet hits an alien. This is achieved by calculating the distance between the bullet and the alien and checking if it is below a certain threshold.





4. CONCLUSION

4. CONCLUSION

The goal of creating a simple, playable game where a player-controlled spaceship defends against alien invaders was successfully achieved. The implementation covers basic game mechanics, providing an engaging experience. Future improvements could include adding multiple levels, different enemy types, and enhancing graphics and sound effects.

```
}
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title= "product
          <div className="row">
            <ProductConsumer>
              {(value) => {
                  console.log(value)
              }}
            </ProductConsumer>
          </div>
        </div>
      </div>
    </React.Fragment>
```

GRETTINGS

MARIO ALVAREZ GRACIA
SIMON BOSQUE GONI