**Project Report on a Game Developed in Pygame**

**Game Title: *Catch the Square***

---

**Submitted By:**

**Name: Mayuresh Mankar**

**Roll No.: 67**

**Class / Semester: TYCS/SEM 5**

**Department of Computer Science**

---

**Submitted To:**

**Faculty Name: Dr.Prabha Kadam**

**Designation: Lecturer**

---

**Academic Year: *2025***

***A project report submitted in partial fulfillment of the requirements for the course in Computer Science***

# Project Report on a Game Developed in Pygame Game

# Title: Catch the Square

---


## Abstract

This project presents a simple yet engaging Python game, **Catch the Square**, developed using the Pygame library. The primary goal of the project is to demonstrate the use of Python in creating interactive games with graphics, collision detection, and scoring mechanisms. The game involves a player-controlled paddle (or AI-controlled) that catches falling squares to earn points. The objective is to score as high as possible by continuously catching the falling objects. The project showcases essential programming concepts such as event handling, game loop structures, collision detection, and optional AI automation. The expected outcome is a fun, minimalistic game that illustrates the fundamentals of game development with Pygame while providing a foundation for future enhancements.


---

## Table of Contents



---

## Introduction

Computer games are an important aspect of both entertainment and learning. With Python and its Pygame library, students and developers can create simple 2D games quickly. This project, **Catch the Square**, demonstrates how Pygame can be used to build a game from scratch that includes player movement, collision detection, scoring, and AI. The objective of the project is to create an interactive and enjoyable experience while learning fundamental game programming concepts.

---

## System Analysis & Design

### Requirements Analysis

- Hardware: Any modern computer with at least 2GB RAM.
- Software: Python 3.x, Pygame library, IDE (e.g., VS Code).

### Game Design

- **Theme:** Simple arcade-style falling object game.
- **Mechanics:** Player moves left/right to catch falling squares.
- **Scoring:** +1 point for each successful catch.
- **AI:** Option to let the computer play by auto-aligning the paddle with the falling square.
- **Data Structures:** Rect objects for player and square, variables for score, and conditionals for collisions.

---

## Implementation

The game was implemented in Python using the Pygame library. The main loop handles user input, AI control, object movement, collision detection, and rendering. The player is represented by a rectangle at the bottom of the screen, while a red square falls from the top. When the paddle collides with the square, the score increases and the square resets to a new random position at the top. The optional AI can automatically control the paddle by aligning it with the square's x-position.

---

## Testing

**Test Cases:**

1. Player moves left/right → Paddle responds correctly.
2. Square falls off screen → Resets at top with random x-position.
3. Collision with paddle → Score increases and square resets.
4. AI mode enabled → Paddle automatically moves to align with square.

All test cases passed successfully.

---

## Results & Discussion

The game successfully demonstrates key features of game programming with Pygame. The player can interactively catch falling squares, and the scoring system works as expected. The AI mode provides an additional layer of functionality, allowing the game to play itself. The performance is smooth at 60 FPS, and the game can run on any system with Python installed.

---

## Conclusion

This project illustrates the potential of Python and Pygame for creating simple, interactive games. **Catch the Square** serves as both a learning tool and a fun minigame, showing how basic programming concepts can be applied to develop functional software. The project achieves its goals of creating an engaging, functional game with optional AI control.

---

## Future Enhancements

- Add levels with increasing difficulty (faster falling squares).
- Add sound effects and background music.
- Implement a menu and game-over screen.
- Save high scores.
- Extend AI to predict future positions more intelligently.

---

## References


1. Pygame Documentation: https://www.pygame.org/docs/ 2.

Python Official Documentation: https://docs.python.org/3/ 3.

Online tutorials and guides for Pygame basics.


---

## Appendices

### Appendix A: Source Code

```
import pygame import
random pygame.init()
# Screen setup
WIDTH, HEIGHT = 600, 400 screen =
pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Catch the Square with AI")

clock = pygame.time.Clock()

# Player setup player = pygame.Rect(WIDTH//2 - 25, HEIGHT
- 40, 50, 20) player_speed = 5

# Falling square square = pygame.Rect(random.randint(0, WIDTH-
20), 0, 20, 20) square_speed = 4

score = 0
font = pygame.font.SysFont(None, 36)

# AI toggle use_ai = True  # set to False if you want to play
manually
```

```
running = True while
running:
    screen.fill((30, 30, 30))

    for event in pygame.event.get():
if event.type == pygame.QUIT:
            running = False

    keys = pygame.key.get_pressed()

    if use_ai:
        # AI automatically moves player towards square
if square.centerx < player.centerx:
            player.x -= player_speed        elif
square.centerx > player.centerx:
            player.x += player_speed
else:
        # Manual control        if
keys[pygame.K_LEFT] and player.left > 0:
            player.x -= player_speed        if
keys[pygame.K_RIGHT] and player.right < WIDTH:
  player.x += player_speed

    # Keep player within bounds     player.x = max(0,
min(WIDTH - player.width, player.x))
```

```
   # Move square     square.y
+= square_speed     if
square.y > HEIGHT:
       square.y = 0        square.x = random.randint(0, WIDTH
- square.width)

   # Collision detection     if
player.colliderect(square):
       score += 1        square.y = 0        square.x =
random.randint(0, WIDTH - square.width)

   # Draw     pygame.draw.rect(screen, (0, 200, 255),
player)     pygame.draw.rect(screen, (255, 80, 80),
square)

   score_text = font.render(f"Score: {score}", True, (255, 255, 255))     mode_text =
font.render("AI Mode" if use_ai else "Manual Mode", True, (180, 180, 180))
screen.blit(score_text, (10, 10))     screen.blit(mode_text, (10, 40))

   pygame.display.flip()
clock.tick(60) pygame.quit()
```
(Game loop handles movement, collision detection, scoring, and optional AI mode.)