Rubric for Final Project: Space Adventure Game

Criteria	Excellent (4)	Good (3)	Satisfactory (2)	Needs Improvement (1)
Game Setup	Pygame initialized, window setup, background color filled correctly.	Pygame initialized, window setup, minor issues with background color.	Pygame initialized with some errors, window setup but incomplete.	Pygame not properly initialized, window setup incorrect.
Game Objects	Classes defined and objects drawn correctly with no errors.	Classes defined with minor errors, objects drawn mostly correctly.	Classes defined but incomplete, objects not drawn correctly.	Classes not properly defined, objects missing or incorrect.
Player Movement	Smooth and responsive movement implemented correctly.	Movement implemented with minor issues.	Movement implemented but not smooth or responsive.	Movement not properly implemented or missing.
Collision Detection	Collision detection accurate and reliable.	Collision detection mostly accurate with minor issues.	Collision detection implemented but not reliable.	Collision detection missing or inaccurate.
Scoring System	Scoring system fully functional and displayed correctly.	Scoring system functional with minor display issues.	Scoring system implemented but not fully functional.	Scoring system missing or incorrect.
Creativity and Design	Game includes creative elements and design, making it engaging.	Game includes some creative elements, design is engaging.	Game includes basic elements, design is simple.	Game lacks creative elements, design is very basic.
Documentation and Code Quality	Code is well-documented and clean.	Code is mostly clean with some documentation.	Code is somewhat clean, lacks documentation.	Code is messy and poorly documented.

Final Game Code

```
import pygame
from pygame.locals import *
import math
import os
import random

# Set SDL audio driver to avoid driver error in console
os.environ['SDL_AUDIODRIVER'] = 'dsp'

# Initialize Pygame
pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Space Adventure")
clock = pygame.time.Clock()

# Colors
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
YELLOW = (255, 255, 0)
RED = (255, 0, 0)
WHITE = (255, 255, 255)

# Spaceship Class
class Spaceship:
    def __init__(self, position):
        self.position = position
        self.color = GREEN

    def draw(self, screen):
        pygame.draw.rect(screen, self.color, (*self.position, 40, 20))

    def move(self, dx, dy):
        self.position[0] += dx
        self.position[1] += dy

# Star and Asteroid Classes
class Star:
    def __init__(self, position):
        self.position = position
        self.color = YELLOW

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, self.position, 8)

class Asteroid:
    def __init__(self, position):
        self.position = position
        self.color = RED

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, self.position, 15)

# Function to detect collision
def detect_collision(obj1, obj2):
    dx = obj1.position[0] - obj2.position[0]
    dy = obj1.position[1] - obj2.position[1]
    distance = math.sqrt(dx**2 + dy**2)
    return distance < 20  # Adjust based on object sizes

# Create game objects
```

```
spaceship = Spaceship([300, 400])
stars = [Star([random.randint(0, 640), random.randint(0, 480)]) for _ in range(5)]
asteroids = [Asteroid([random.randint(0, 640), random.randint(0, 480)]) for _ in
range(3)]

# Initialize score
score = 0

# Main game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == QUIT:
            running = False

    keys = pygame.key.get_pressed()
    if keys[K_LEFT] and spaceship.position[0] > 0:
        spaceship.move(-5, 0)
    if keys[K_RIGHT] and spaceship.position[0] < 600:
        spaceship.move(5, 0)
    if keys[K_UP] and spaceship.position[1] > 0:
        spaceship.move(0, -5)
    if keys[K_DOWN] and spaceship.position[1] < 460:
        spaceship.move(0, 5)

    # Check for collisions with stars
    for star in stars[:]:
        if detect_collision(spaceship, star):
            stars.remove(star)
            score += 10
            print(f"Collected a star! Score: {score}")

    # Check for collisions with asteroids
    for asteroid in asteroids:
        if detect_collision(spaceship, asteroid):
            score -= 5
            print(f"Hit an asteroid! Score: {score}")

    screen.fill(BLACK)
    spaceship.draw(screen)
    for star in stars:
        star.draw(screen)
    for asteroid in asteroids:
        asteroid.draw(screen)

    # Display the score
    font = pygame.font.Font(None, 36)
    score_text = font.render(f"Score: {score}", True, WHITE)
    screen.blit(score_text, (10, 10))

    pygame.display.flip()
    clock.tick(30)

pygame.quit()
```