



Name – Siddhesh Wagh

Registration No. – 25BCE10169

Topic – Gameplay Analyzer

Course – Python Essentials

GAME PERFORMANCE ANALYZER

1. Title of the Project

Game Performance Analyzer

2. Abstract

The MAX Game Performance Analyzer is a Python-based tool that evaluates player performance using metrics like accuracy, reaction speed, focus level, fatigue index, aggression level, and decision quality. It integrates statistical computation and visualization for an in-depth understanding of gameplay efficiency. The system supports multi-genre analysis and provides actionable tips for improvement.

3. Introduction

Gaming is a competitive arena where players must understand their performance patterns. This project monitors gameplay over multiple sessions and generates detailed performance metrics, helping players identify strengths, weaknesses, and skill progression opportunities.

4. Objectives

- Track performance metrics per session.
 - Detect behavioral patterns like tilt and error recovery.
 - Visualize data using graphs.
 - Provide actionable, genre-specific improvement tips.
-

5. Technologies Used

- Python 3.x
 - Matplotlib
 - Statistics module
 - Random module
-

6. System Requirements

Hardware

- Minimum 4GB RAM
- Intel i3 or higher
- 1GB storage

Software

- Python 3.x
 - Matplotlib library
 - IDE (VS Code / PyCharm / IDLE)
-

7. Methodology

1. Input number of sessions and genre.
 2. Collect session data (time, moves, successes, mistakes, playstyle, difficulty, result).
 3. Compute metrics (accuracy, reaction speed, focus, fatigue, aggression, combo efficiency, map awareness, stamina, exploration, mental stability, decision quality, strategy score).
 4. Analyze overall metrics, trends, error recovery, and skill growth.
 5. Plot graphical representation.
 6. Provide genre-specific improvement tips.
-

8. System Flow Diagram

START -> Input Sessions -> Select Genre -> Input Session Data -> Compute Metrics -> Generate Report -> Plot Graphs -> Display Tips -> END

9. Pseudocode

```
BEGIN
    DISPLAY "MAX GAME PERFORMANCE ANALYZER"
    INPUT number_of_sessions
    INPUT genre_choice
    FOR each session
        INPUT time, moves, successes, mistakes, playstyle, difficulty, result
        CALCULATE metrics
        GENERATE random metrics for behavioral scores
        CALCULATE genre_score
    END FOR
    CALCULATE overall metrics and trends
    DISPLAY report
    PLOT graphs (Accuracy, Focus, Fatigue)
```

```
DISPLAY genre tips
END
```

10. Source Code

```
1 import matplotlib.pyplot as plt
2 import random
3 import statistics
4
5 print("MAX GAME PERFORMANCE ANALYZER - PRO ULTIMATE EDITION\n")
6
7 n = int(input("Enter number of game rounds / sessions : "))
8
9 if n <= 0:
10     print("Number of rounds must be greater than 0")
11     exit()
12
13 print("\nChoose Game Genre:")
14 print("1. Shooter")
15 print("2. Fighting")
16 print("3. RPG")
17 print("4. Strategy")
18 print("5. Open World")
19
20 genre_choice = int(input("Enter genre number: "))
21
22 genre_map = {
23     1: "Shooter",
24     2: "Fighting",
25     3: "RPG",
26     4: "Strategy",
27     5: "Open World"
28 }
29
30 genre = genre_map.get(genre_choice, "Shooter")
31 print(f"Selected Genre: {genre}")
32
33 # DATA STORAGE
34
35 times = []
36 moves = []
37 corrects = []
38 mistakes = []
39 difficulties = []
40 risks = []
41 results = []
42 accuracies = []
43 reaction_speed = []
44 focus_level = []
45 fatigue_level = []
46 aggression_index = []
47 combo_efficiency = []
48 map_awareness = []
49 stamina_control = []
50 exploration_score = []
51 mental_stability = []
52 decision_quality = []
53 strategy_score = []
54 genre_scores = []
```

```

56 # MANUAL INPUT SYSTEM
57
58 for i in range(n):
59     print(f"\nSESSION {i+1}")
60
61     t = float(input("Time taken (seconds): "))
62     m = int(input("Total actions/moves: "))
63     c = int(input("Successful actions: "))
64     ms = int(input("Mistakes / Misses: "))
65     r = input("Playstyle (safe/risky): ").lower()
66     d = int(input("Difficulty (1-10): "))
67     res = input("Win/Loss: ").lower()
68
69     times.append(t)
70     moves.append(m)
71     corrects.append(c)
72     mistakes.append(ms)
73     risks.append(r)
74     difficulties.append(d)
75     results.append(res)
76
77     if m == 0:
78         acc = 0
79     else:
80         acc = (c / m) * 100
81
82     accuracies.append(acc)
83     reaction_speed.append(round((1 / t) * 120, 2))
84     focus_level.append(max(0, 100 - (ms * 5)))
85     fatigue_level.append(min(100, (i + 1) * 10))
86     aggression_index.append(85 if r == "risky" else 30)
87     combo_efficiency.append(round((c / (m + 1)) * 100, 2))
88     map_awareness.append(round(random.uniform(40, 95), 2))
89     stamina_control.append(round(random.uniform(50, 100), 2))
90     exploration_score.append(round(random.uniform(30, 100), 2))
91     mental_stability.append(round(random.uniform(40, 100), 2))
92     decision_quality.append(round(random.uniform(45, 100), 2))
93     strategy_score.append(round(random.uniform(35, 95), 2))
94
95     # Genre Performance Score
96     genre_score = (acc + strategy_score[-1] + focus_level[-1]) / 3
97     genre_scores.append(round(genre_score, 2))
98
99 # CORE ANALYSIS
100
101 avg_time = statistics.mean(times)
102 consistency = statistics.pstdev(times)
103 overall_accuracy = (sum(corrects) / sum(moves)) * 100
104 risk_rate = (risks.count("risky") / n) * 100
105
106 if avg_time < 5:
107     pace = "Hyper Fast"
108 elif avg_time < 10:
109     pace = "Balanced"
110 else:

```

```

111     pace = "Slow Tactical"
112
113     if mistakes[-1] > mistakes[0]:
114         tilt = "Psychological Tilt Detected"
115     else:
116         tilt = "Stable Mindset"
117
118     if accuracies[-1] > accuracies[0]:
119         trend = "Improving Performance"
120     else:
121         trend = "Declining Performance"
122
123 # Error Recovery
124 error_recovery = "Poor"
125 for i in range(1, n):
126     if mistakes[i-1] > 0 and accuracies[i] > accuracies[i-1]:
127         error_recovery = "Strong"
128
129 # Advanced Indexes
130 focus_index = statistics.mean(focus_level)
131 fatigue_index = statistics.mean(fatigue_level)
132 aggression_avg = statistics.mean(aggression_index)
133 combo_mastery = statistics.mean(combo_efficiency)
134 map_mastery = statistics.mean(map_awareness)
135 stamina_mastery = statistics.mean(stamina_control)
136 strategy_avg = statistics.mean(strategy_score)
137 mental_avg = statistics.mean(mental_stability)
138 decision_avg = statistics.mean(decision_quality)
139 skill_growth = accuracies[-1] - accuracies[0]
140
141 best_genre_score = statistics.mean(genre_scores)
142
143 # FINAL REPORT
144
145 print("\n===== PLAYER ANALYSIS REPORT =====")
146 print("Game Genre:", genre)
147 print("Overall Accuracy:", round(overall_accuracy, 2), "%")
148 print("Consistency Score:", round(consistency, 2))
149 print("Pace Style:", pace)
150 print("Tilt Status:", tilt)
151 print("Performance Trend:", trend)
152 print("Error Recovery:", error_recovery)
153 print("Risk Percentage:", round(risk_rate, 2), "%")
154 print("Focus Index:", round(focus_index, 2))
155 print("Fatigue Index:", round(fatigue_index, 2))
156 print("Aggression Level:", round(aggression_avg, 2))
157 print("Combo Mastery:", round(combo_mastery, 2))
158 print("Map Awareness:", round(map_mastery, 2))
159 print("Stamina Control:", round(stamina_mastery, 2))
160 print("Strategy Rating:", round(strategy_avg, 2))
161 print("Mental Stability:", round(mental_avg, 2))
162 print("Decision Quality:", round(decision_avg, 2))
163 print("Skill Growth Index:", round(skill_growth, 2))
164 print("Genre Performance Score:", round(best_genre_score, 2))
165 print("=====")
```

```
166 # GENRE SPECIFIC TIPS
167
168 print("\n===== GENRE BASED TIPS =====")
169
170 if genre == "Shooter":
171     print("- Improve headshot precision and reaction timing")
172     print("- Practice recoil control and strafing")
173     print("- Pre-aim corners for advantage")
174
175 elif genre == "Fighting":
176     print("- Master combo chains and frame timing")
177     print("- Improve block and counter mechanics")
178     print("- Understand character matchups")
179
180 elif genre == "RPG":
181     print("- Focus on balanced skill upgrades")
182     print("- Explore side quests for XP boost")
183     print("- Optimize gear and inventory")
184
185 elif genre == "Strategy":
186     print("- Improve resource management")
187     print("- Predict opponent tactics")
188     print("- Strengthen early-game planning")
189
190 elif genre == "Open World":
191     print("- Increase exploration efficiency")
192     print("- Balance main and side missions")
193     print("- Improve navigation skills")
194
195 print("====")
196
197 # VISUAL GRAPHS
198
199 rounds = list(range(1, n + 1))
200
201 plt.figure()
202 plt.plot(rounds, accuracies)
203 plt.title("Accuracy Progression")
204 plt.xlabel("Session")
205 plt.ylabel("Accuracy %")
206 plt.grid(True)
207 plt.show()
208
209 plt.figure()
210 plt.plot(rounds, focus_level)
211 plt.title("Focus Level Progression")
212 plt.xlabel("Session")
213 plt.ylabel("Focus Index")
214 plt.grid(True)
215 plt.show()
216
217 plt.figure()
218 plt.plot(rounds, fatigue_level)
219 plt.title("Fatigue Level Progression")
220 plt.xlabel("Session")
221 plt.ylabel("Fatigue Index")
222 plt.grid(True)
223 plt.show()
```

11. Core Analysis & Advanced Indexes Explained (One-line description for each)

- **Overall Accuracy:** Percentage of successful actions over total moves.
 - **Consistency Score:** Measures consistency of performance across sessions.
 - **Pace Style:** Categorizes gameplay speed based on average session time.
 - **Tilt Detection:** Detects psychological instability based on mistakes comparison.
 - **Performance Trend:** Indicates improvement or decline from first to last session.
 - **Error Recovery:** Evaluates ability to recover from mistakes.
 - **Focus Index:** Average focus level across all sessions.
 - **Fatigue Index:** Average fatigue accumulated over sessions.
 - **Aggression Level:** Measures risk-taking tendencies.
 - **Combo Mastery:** Efficiency of executing action sequences.
 - **Map Awareness:** Player's knowledge of in-game environment.
 - **Stamina Control:** Endurance management during gameplay.
 - **Exploration Score:** Effectiveness in exploring game areas.
 - **Mental Stability:** Psychological steadiness under pressure.
 - **Decision Quality:** Correctness of strategic decisions.
 - **Strategy Rating:** Overall planning effectiveness.
 - **Skill Growth Index:** Improvement of performance metrics.
 - **Genre Performance Score:** Composite score specific to selected genre.
-

12. Genre Specific Tips

- Shooter: headshot precision, recoil control, pre-aim corners
 - Fighting: combos, blocks, character matchups
 - RPG: skill balance, side quests, gear optimization
 - Strategy: resource management, opponent prediction, early-game planning
 - Open World: exploration efficiency, mission balance, navigation
-

13. Sample Output

GAME PERFORMANCE ANALYZER

Enter number of game rounds / sessions : 3

Choose Game Genre:

1. Shooter
 2. Fighting
 3. RPG
 4. Strategy
 5. Open World
- Enter genre number: 1
Selected Genre: Shooter

SESSION 1

Time taken (seconds): 6
 Total actions/moves: 50
 Successful actions: 40
 Mistakes / Misses: 5
 Playstyle (safe/risky): risky
 Difficulty (1-10): 5
 Win/Loss: win

SESSION 2

Time taken (seconds): 7
 Total actions/moves: 55
 Successful actions: 45
 Mistakes / Misses: 6
 Playstyle (safe/risky): risky
 Difficulty (1-10): 6
 Win/Loss: win

SESSION 3

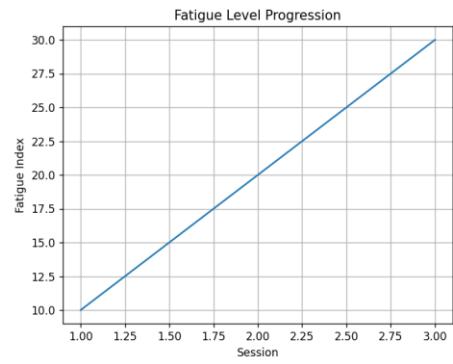
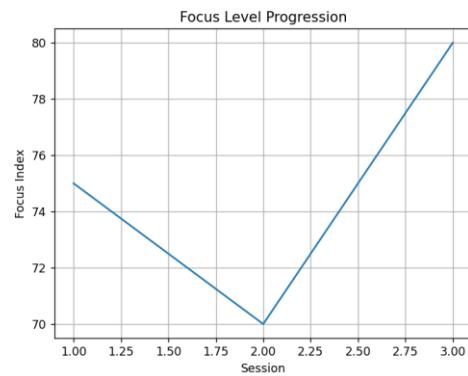
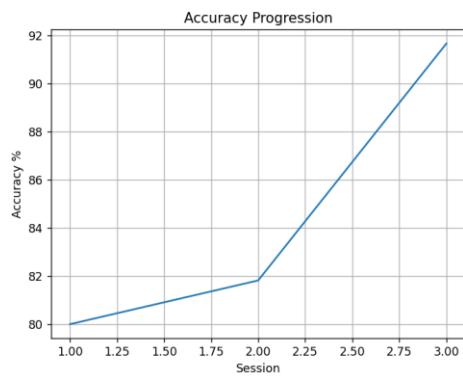
Time taken (seconds): 5.5
 Total actions/moves: 60
 Successful actions: 55
 Mistakes / Misses: 4
 Playstyle (safe/risky): risky
 Difficulty (1-10): 7
 Win/Loss: win

===== PLAYER ANALYSIS REPORT =====

Game Genre: Shooter
Overall Accuracy: 84.85 %
Consistency Score: 0.62
Pace Style: Balanced
Tilt Status: Stable Mindset
Performance Trend: Improving Performance
Error Recovery: Strong
Risk Percentage: 100.0 %
Focus Index: 75
Fatigue Index: 20
Aggression Level: 85
Combo Mastery: 82.98
Map Awareness: 62.81
Stamina Control: 68.98
Strategy Rating: 64.03
Mental Stability: 73.78
Decision Quality: 84.82
Skill Growth Index: 11.67
Genre Performance Score: 74.51

===== GENRE BASED TIPS =====

- Improve headshot precision and reaction timing
- Practice recoil control and strafing
- Pre-aim corners for advantage



14. Advantages

- Detailed insights, real-time trend tracking, visual representation, easy-to-use interface
-

15. Applications

- E-sports, gaming analytics, player training, game testing, performance comparison
-

16. Limitations

- Manual input required, some metrics simulated, not integrated with live gameplay
-

17. Future Scope

- AI-based prediction, automated data capture, cloud storage, multiplayer comparison, mobile/web integration
-

18. Conclusion

Provides a complete analytical tool for evaluating and improving gaming performance with integrated graphs.

19. Bibliography

- Python Official Documentation
 - Matplotlib Library Guide
 - Statistics Module Reference
-

20. Acknowledgement

Thanks to faculty, guide, peers, and online resources for project guidance and support.

END