QuizGame Client-Server with Unity

Tommaso Spadaro

Indice

1 Description

I have designed and developed a quiz-game in C dotnet through client-server paradigm based on TCP socket. The application let the players to connect to the server, receive questions, send answer and compete each other in real time.
The system include turn management, score, answer time and correctness of the answers.

1.1 System Architecture

- **Server**: Implemented as Singleton(`Party`), manage game status and client-players, selection questions and evaluated the answers.
- **Client**: Each player can connect to the server writing a name-tag, receive the questions and send answers within time limit.
- **Communication**: Based on `TcpClient` and `TcpListener` through `NetworkStream` and UTF-8 codification.

1.2 Main Component

- `Player`: Class with all information about the connected players: name, score, time of answering and correctness.
- `Party`: Manage the game logic, handles multi-thread synchronization: send questions, receive answers and calculate the classification.
- `DataBaseQuiz`: Singleton that gives random questions based on category and difficulty, and verify the answers.
- `Stopwatch`: Class to measure the time used by each player to answer.

2 Game Logic

1. Clients connect to the server and send their names.
2. The host (via console) chooses the category and difficulty, then starts the game.
3. All clients receive the same question with 3 answer options.
4. A shared 15-second timer begins.
5. Each client sends their answer, or they are marked as "absent" if the time expires.
6. The server calculates the score:
   - +1 point for a correct answer.
   - +1 extra point for the first player to answer correctly.
7. At the end of the turn, each player receives a personalized evaluation of their result.

2.1 Technologies Used

- **Language**: C#
- **Libraries**: `System.Net.Sockets`, `System.Threading.Tasks`, `System.Text`, `LINQ`
- **Paradigms**: Asynchronous programming, Singleton, multithreading with `Task` and `CancellationToken`

2.2 Project Strengths

- Secure concurrent management of multiple clients.
- Synchronization between game time and user input.
- Separation between game logic and client interface.
- Complete networking experience with stream management, threads, and real-time interaction.

3 Unity Client Integration

To make the quiz experience more engaging and accessible on mobile devices, I developed a client in **Unity**, connected to the TCP server via sockets, which graphically manages the questions, answers, and user interaction.

3.1 User Interface

Upon arrival of a new question, the client displays the question text at the top of the screen and three answer alternatives within colored circles: pink, green, and yellow. Each alternative is positioned horizontally on the screen. The user controls a blue hexagon using the keyboard's directional arrows and selects an answer by moving it into the desired circle.

3.2 Time Management and Feedback

Each question has a time limit. When the time expires, a message indicates whether the user answered correctly or incorrectly. If no answer was provided in time, the message "Time expired!" is displayed.

4 Visual Examples

4.1 Correct Answer

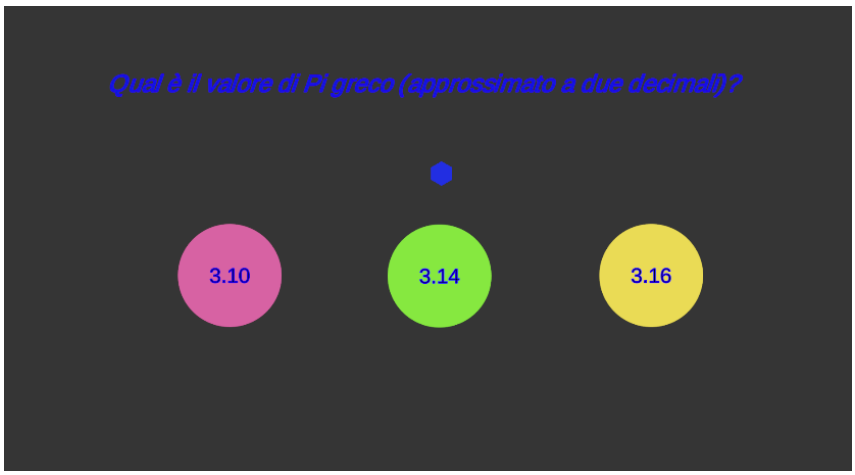

Figura 1: Initial question with player in central position.

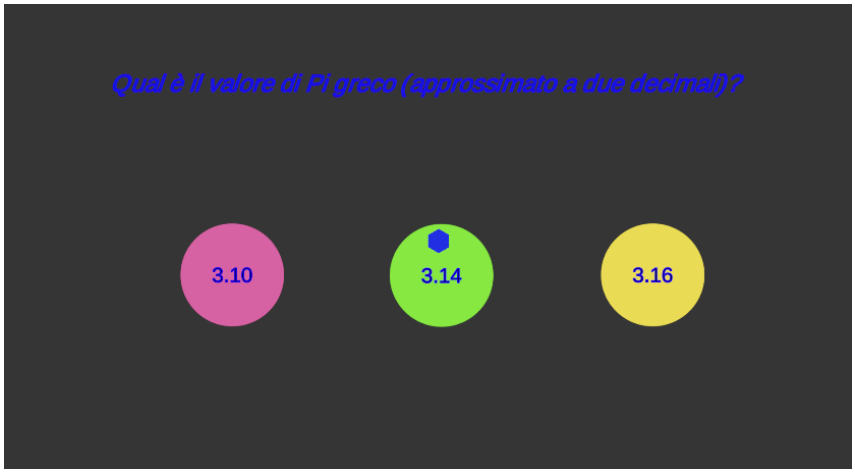

Figura 2: The player moves to the circle containing the chosen answer.

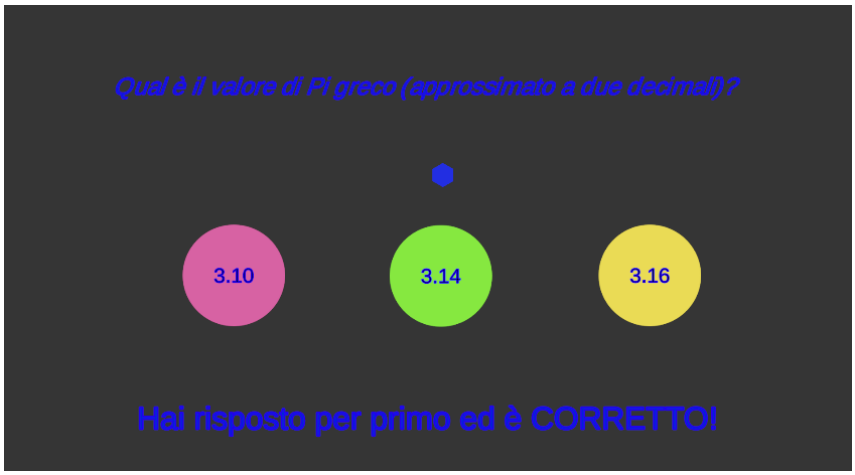

Figura 3: The player receives positive feedback for their answer.

4.2 Incorrect Answer

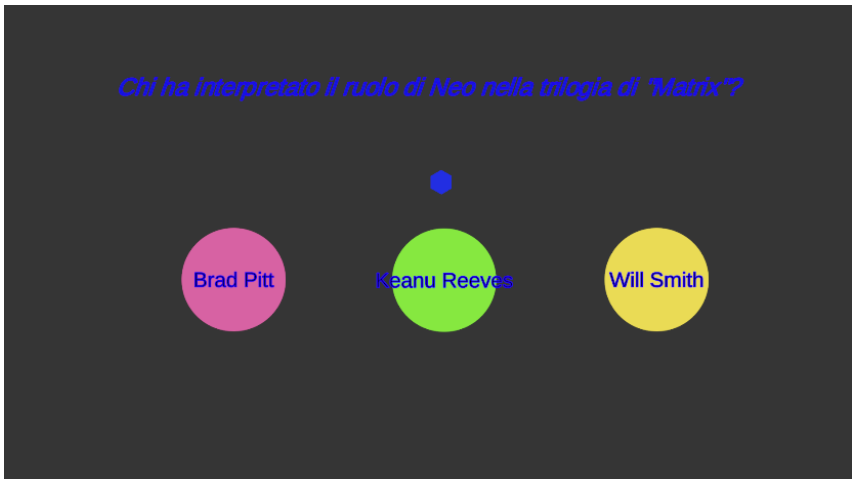

Figura 4: Initial question with player in central position.

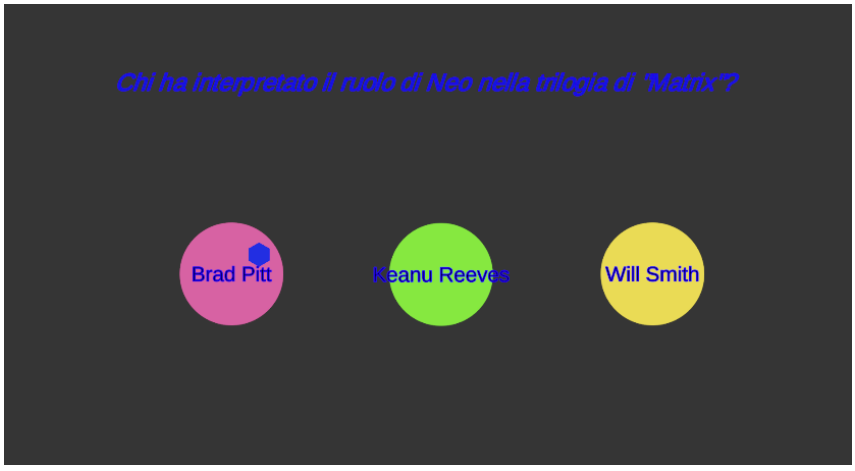

Figura 5: The player moves to the circle containing the chosen answer.

Figura 6: The player receives negative feedback for their answer.

4.3 Time Expired

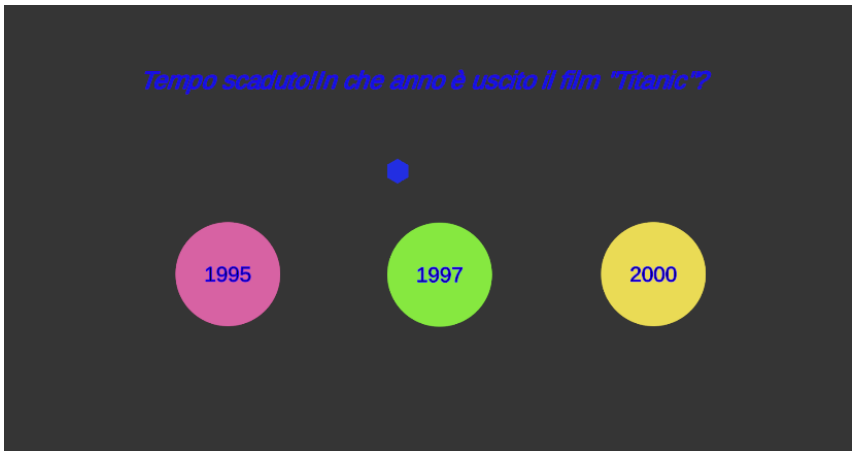

Figura 7: Time expiration with a new question displayed.

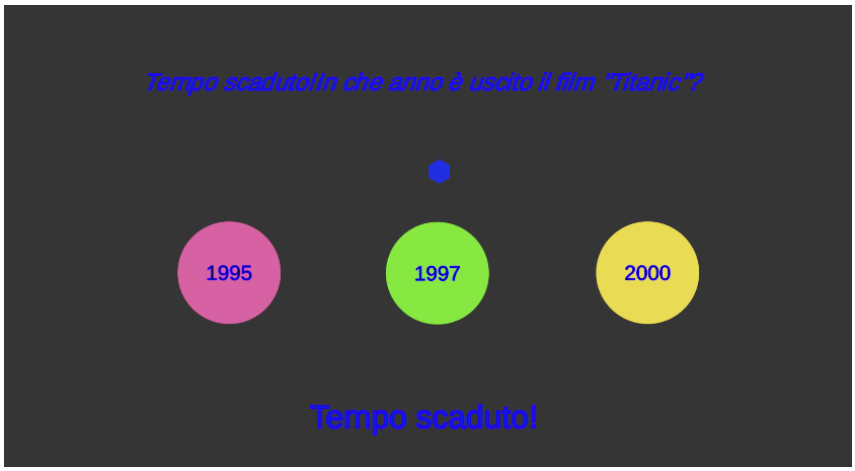

Figura 8: Final message communicating that time has expired.