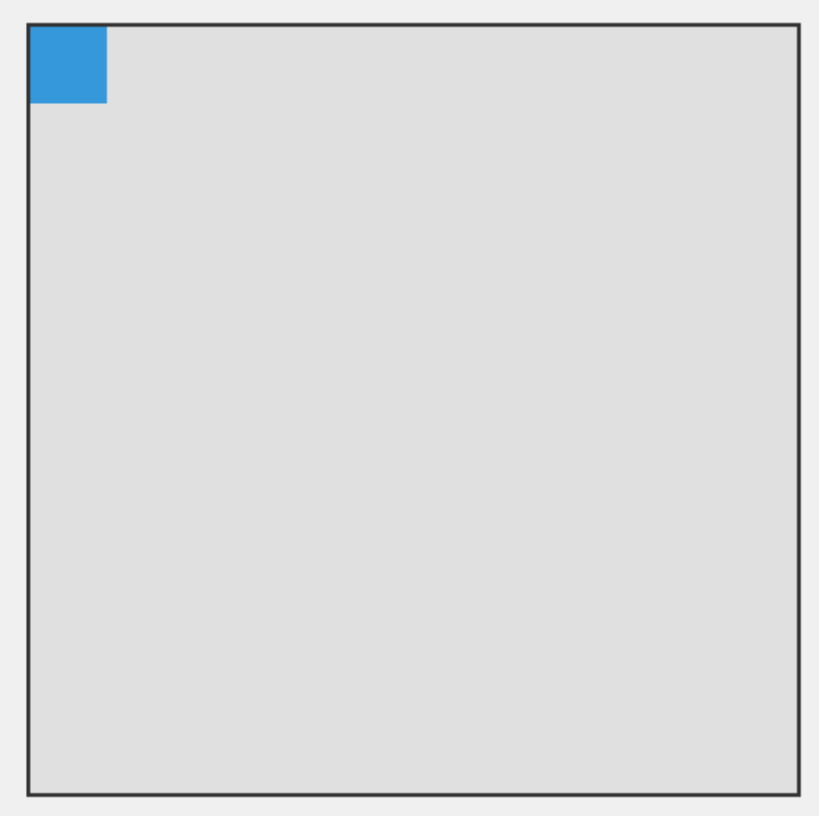
Documentazione tecnica “Gioco del quadrato”

Diego Tempesta, Rebecca Truden, Erika Saddi



Indice della Documentazione del Software

1. Introduzione

1.1 Scopo del Software

Il "Gioco del Quadrato" è un applicazione web interattiva che sfida l'utente a muovere un quadrato all'interno di un'area di gioco utilizzando i tasti direzionali. L'obiettivo è raggiungere l'angolo in basso a destra prima che il tempo scada.

1.2 Descrizione Generale del Software

Il software è implementato in HTML, CSS e JavaScript. È progettato per funzionare in un browser moderno e offre un'interfaccia semplice con una grafica minimale. Le funzionalità principali includono:

- Movimenti del quadrato tramite tastiera.
- Timer per la durata del gioco.
- Controllo delle collisioni e delle condizioni di vittoria.

1.3 Obiettivi della Documentazione

Fornire una descrizione esaustiva del software, inclusi l'architettura, il design, i test e la manutenzione, per supportare sviluppatori e utenti finali.

1.4 Panoramica del Documento

Questa documentazione è suddivisa in sezioni che descrivono l'architettura, le proprietà, il design, i test, il deployment e la manutenzione del software, con esempi di codice e diagrammi utili.

2. Architettura del Software

2.1 Descrizione Generale dell'Architettura

Il software è basato su una semplice architettura client-side. La struttura principale è composta da:

- **HTML**: struttura del layout.
- **CSS**: stile e layout visivo.
- **JavaScript**: logica interattiva.

2.2 Componenti Principali

- **Interfaccia utente**: elementi HTML per l'area di gioco, il quadrato, il timer e i messaggi di stato.
- **Motore di gioco**: script JavaScript che gestisce il movimento, il timer e le condizioni di vittoria/sconfitta.

2.3 Flusso di Dati

1. L'utente interagisce con la tastiera.
2. Gli eventi `keydown` aggiornano le coordinate del quadrato.
3. La posizione del quadrato viene verificata per condizioni di vittoria.
4. Un timer decrementa il tempo rimanente, gestendo lo stato di fine partita.

2.4 Diagrammi UML

- **Diagramma delle classi**: Non applicabile per questo progetto.
- **Diagramma di sequenza**: Disponibile su richiesta.
- **Diagramma di stato**: Disponibile su richiesta.

3. Analisi delle Proprietà

3.1 Robustezza

3.1.1 Definizione di Robustezza nel Contesto del Software

La robustezza si riferisce alla capacità del software di gestire input e stati inattesi senza comportamenti anomali.

3.1.2 Gestione degli Errori

Il codice gestisce i bordi dell'area di gioco per prevenire movimenti fuori limite.

3.1.3 Test di Robustezza

- Simulazione di input rapidi e continui.
- Movimenti verso i bordi dell'area di gioco.

3.1.4 Resilienza del Software

Il gioco continua a funzionare correttamente anche con input errati (es. tasti non direzionali).

3.1.5 Codice Documentato per la Gestione degli Errori

```
if (e.key === 'ArrowUp' && playerPos.y > 0) {
    playerPos.y -= moveSpeed;
}
```

3.2 Usabilità

3.2.1 Definizione di Usabilità nel Contesto del Software

L'usabilità valuta la facilità d'uso e l'efficacia dell'interfaccia utente.

3.2.2 Interfaccia Utente (UI) e User Experience (UX)

Un'area di gioco ben definita e messaggi di stato chiari migliorano l'esperienza.

3.2.3 Accessibilità

Funziona su qualsiasi browser moderno senza dipendenze aggiuntive.

3.2.4 Feedback degli Utenti

I messaggi di stato indicano chiaramente le condizioni di vittoria/sconfitta.

3.2.5 Codice Documentato per la UI/UX

```
statusDisplay.textContent = 'Hai vinto!';
```

3.3 Portabilità

3.3.1 Definizione di Portabilità nel Contesto del Software

La capacità del software di funzionare su piattaforme diverse senza modifiche.

3.3.2 Compatibilità tra Piattaforme

Il gioco è compatibile con browser moderni (Chrome, Firefox, Safari).

3.3.3 Dipendenze Esterne e Gestione della Configurazione

Non utilizza librerie esterne.

3.3.4 Strategie di Testing Cross-Platform

Funzionante solo su browser da PC.

3.3.5 Codice Documentato per la Portabilità

Non richiesto per questo progetto.

4. Design del Software

4.1 Principi di Design Adottati

- Semplicità.
- Responsività.
- Feedback immediato all'utente.

4.2 Pattern di Design Utilizzati

Nessuno specifico.

4.3 Scelte Architetturali e Motivi

Uso di JavaScript nativo per evitare dipendenze.

5. Documentazione del Codice

5.1 Struttura del Codice Sorgente

- `index.html`: struttura.
- `style`: stili.
- `script`: logica di gioco.

5.2 Classi e Metodi Principali

Nessuna classe, solo funzioni semplici.

5.3 Commenti e Note Importanti nel Codice

Struttura HTML: Commenti per descrivere il ruolo di ciascun elemento (contenitore di gioco, timer, status, pulsante).

CSS: Spiegazione degli stili principali e dell'uso di `overflow: hidden` per limitare il movimento del quadrato.

JavaScript:

- Dichiarazione delle variabili e la loro funzione.
- Spiegazioni delle condizioni nei metodi `startGame` e `movePlayer`.
- Dettagli sull'aggiornamento della posizione del quadrato e sulla gestione del timer.

5.4 Esempi di Utilizzo del Codice

N/A.

5.5 Snippet di Codice per le Proprietà Analizzate

Esempio per la robustezza:

```
if (timeLeft <= 0) {
    clearInterval(timer);
    statusDisplay.textContent = 'Tempo scaduto! Hai perso!';
}
```

6. Testing

6.1 Strategia di Testing

Test manuale.

6.2 Test Unitari

N/A.

6.3 Test di Integrazione

Movimento e timer testati insieme.

6.4 Test di Performance e Stress

Testato con input continui.

6.5 Test Cross-Platform

Non disponibile al di fuori di dispositivi desktop.

6.6 Risultati dei Test e Report

Funzionamento corretto.

7. Deployment e Portabilità

7.1 Guida al Deployment

Caricare i file su un server web.

7.2 Ambienti di Produzione e Sviluppo

Identici.

7.3 Esempi di Configurazione

N/A.

7.4 Considerazioni sulla Portabilità durante il Deployment

Assicurarsi che il server supporti file statici.

8. Manutenzione del Software

8.1 Strategie di Manutenzione

Aggiornamenti periodici.

8.2 Aggiornamenti del Software

Prevedere nuove funzionalità.

8.3 Bug Fixing e Refactoring

Ottimizzare il codice.

8.4 Backup e Recupero dei Dati

Non ci sono dati da salvare.

9. Conclusioni

9.1 Sintesi delle Proprietà Analizzate

Progetto semplice e robusto.

9.2 Suggerimenti Futuri per il Miglioramento

Aggiungere livelli o sfide.

9.3 Lezioni Apprese durante lo Sviluppo

Importanza di feedback immediato.

10. Appendici

10.1 Glossario

- **Quadrato**: elemento giocabile.
- **Area di gioco**: spazio delimitato.

10.2 Riferimenti e Bibliografia

Nessuno.

10.3 Link Utili e Risorse Esterne

Documentazione JavaScript su MDN Web Docs.