Progetto Reti di Calcolatori

Anno Accademico 2020-2021

Sessione estiva

Euclide

Daniele Mariella
261064

Docente: Prof. Antonio Della Selva

Introduzione

Come progetto d'esame ho scelto di implementare un gioco di tipo Client/Server, per approfondire l'argomento affrontato durante il corso.

Il nome del progetto è Euclide, un gioco 1 vs 1 dove i giocatori si sfidano all'interno di uno spazio circoscritto nel quale ogni utente realizza un punto spingendo delle palline dinamiche che si scontrano tra di loro all'interno dello spazio, verso le diverse aperture disponibili.

Il programma prevede poi la possibilità di avere più di due giocatori che giocano in modo simultaneo in stanze differenti.

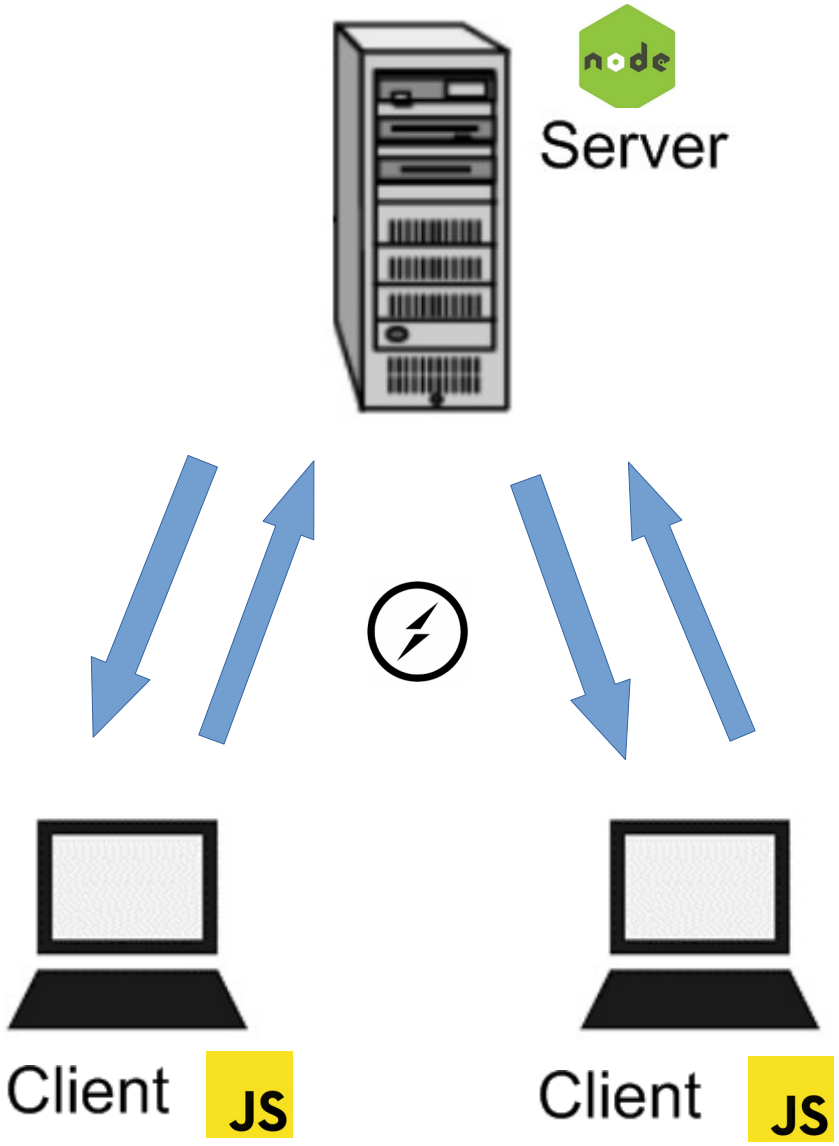
Per questo progetto si presta molto bene l’utilizzo di Node.js affiancato dall’utilizzo di Javascript, HTML e CSS.

Per la distribuzione ho utilizzato Heroku.

Il gioco è disponibile al seguente indirizzo:

https://euclidegioco.herokuapp.com/

Struttura client/server



Il paradigma client/server prevede la presenza di diversi clienti che richiedono servizi.
Senza il server questo paradigma non funzionerebbe poiché è lui il principale supervisore che si occupa dello smistamento e dell'indirizzamento delle richieste dei clienti. La comunicazione tra clienti è quindi una comunicazione di tipo indiretto, ovvero tutto passa attraverso il server prima di arrivare a destinazione.

Questa configurazione client-server rappresenta la struttura del progetto.
Sono coinvolti due processi cioè due programmi in esecuzione, uno sulla macchina client e uno sulla macchina server.

Il client e il server si inviano e ricevono messaggi attraverso canali di comunicazione, eseguiti tramite l'utilizzo della libreria di socket.io.

Progetto implementato dall'ambiente runtime Javascript → Node.js

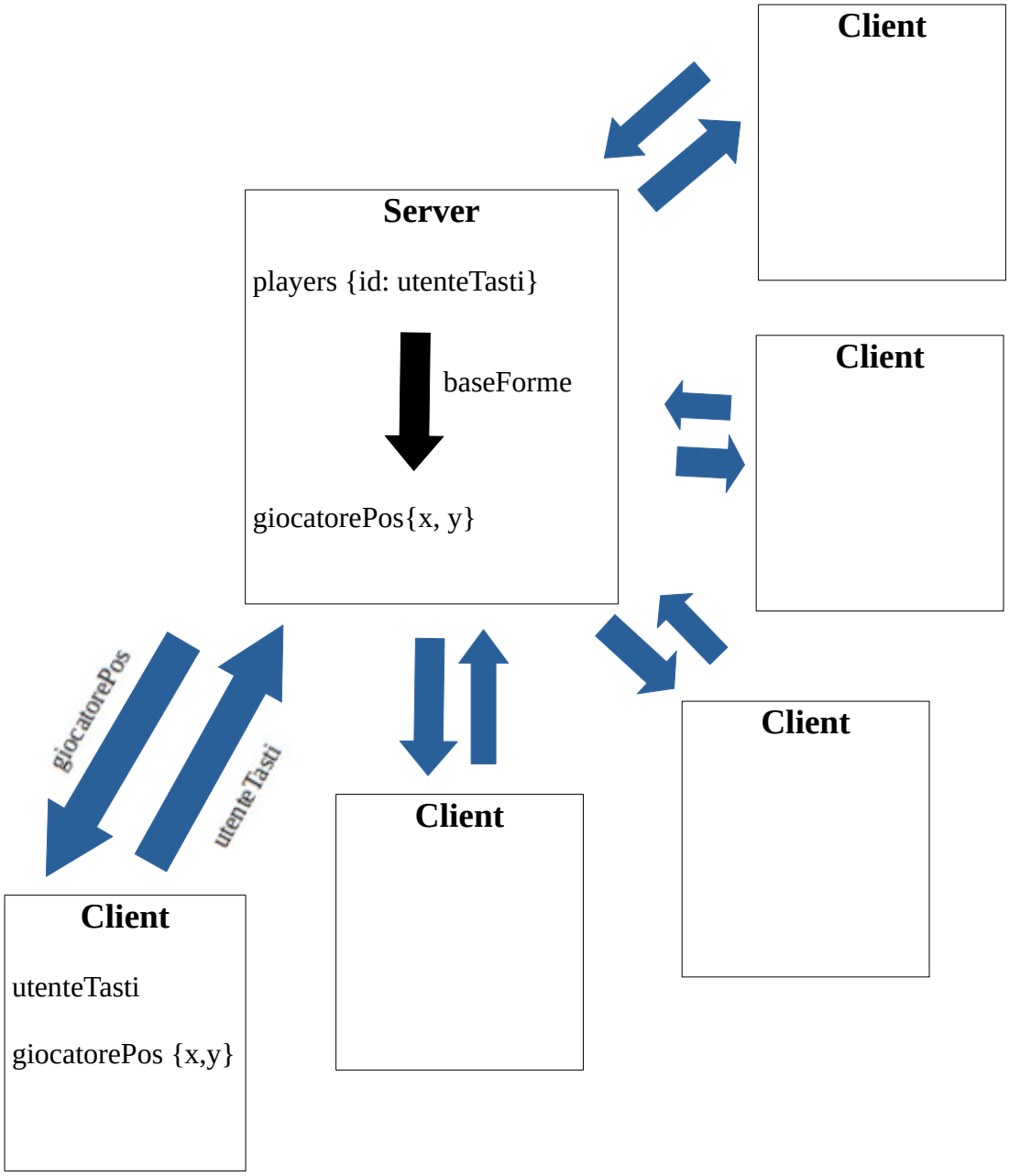
Trasferimento di informazioni tra client e server → utilizzo di 2 funzioni base di socket.io:

- emit() → utilizzato per inviare informazioni
- on() → utilizzato per ricevere informazioni

entrambe le funzioni hanno un "event name" come primo argomento, mentre il secondo argomento è opzionale.

Il client è solamente connesso al server e dunque emette informazioni (data) al server ma il server è connesso con ogni client e ha dunque diverse opzioni di emissione.

Modello lato Server



L'idea è che tutta la parte fisica (baseForme) del gioco venga calcolata nel lato server. Il Client invia "utenteTasti" ma il server invece di fare broadcast di questi valori, svolge il lavoro di calcolare la posizione del giocatore basata su questi valori.
Invia poi la posizione (giocatorePos) al client.

In questo caso tutto quello che succede nel lato client è rappresentare gli oggetti nelle posizioni ricevute.

Questo è il motivo per il quale ho incluso il codice di baseForme anche nel lato Server, per simulare gli urti e tenere traccia delle posizioni. (freccia nera)

Per il futuro, dato che questo non è il modo più elegante di rappresentare la parte fisica nel lato server l'idea è di creare un npm package al di fuori del progetto cosicché basteranno poche linee di codice per importare come modulo questa parte, evitando ridondanza di codice.

Le funzioni principali del server sono quelle di far connettere il client, la disconnessione dei giocatori e dunque la gestione della partita.

Client

Il client nel gioco è un utente/giocatore.

Per connettersi il client necessita di inserire il suo username, e in seguito cliccando il pulsante "Start", si unirà alla partita in una determinata stanza.

Passi del Client

Il client dopo aver stabilito la connessione ed essersi unito alla partita, attenderà che il secondo giocatore si connetta così da poter iniziare.
Quando un giocatore perderà, non avverrà la disconnessione definitiva perché potrà giocare ancora fino a quando vorrà.

Per disconnettersi definitivamente, dovrà chiudere la pagina o ricaricarla.

Presentazione → Euclide

Regole del gioco

1. Ogni giocatore ha una sua Pallina (verde o rossa).

2. Le 4 palline da spingere hanno valore di 1 punto.

3. Le 4 palline con le quali si può fare punto hanno rispetto alla Pallina utente:

   - ✔ un attrito maggiore;
   - ✔ velocità minore;
   - ✔ dimensione più piccola.

4. I giocatori si sfidano all'interno dello spazio spingendo le 4 palline ostacolandosi a vicenda.

5. Dopo la realizzazione di un punto la pallina (non utente) sarà ripristinata nella posizione di partenza.

6. Il giocatore vince quando avrà raggiunto un punteggio pari a 4 prima dell'avversario.

7. Se una delle 4 palline verrà spinta da se stessi, verso una delle 2 aperture del proprio lato, il punto sarà dato all'avversario.

Per l'implementazione si è suddiviso il problema principale in sotto-problemi:

- Implementazione interfaccia grafica
- Gestione Client
- Gestione Server
- Implementazione regole del gioco

Implementazione grafica

La schermata iniziale, rappresenta la pagina in cui l'utente (client) dovrà inserire il proprio nickname personalizzato che sarà poi riportato nella schermata successiva. Esso prevede la dimensione massima di 10 caratteri.

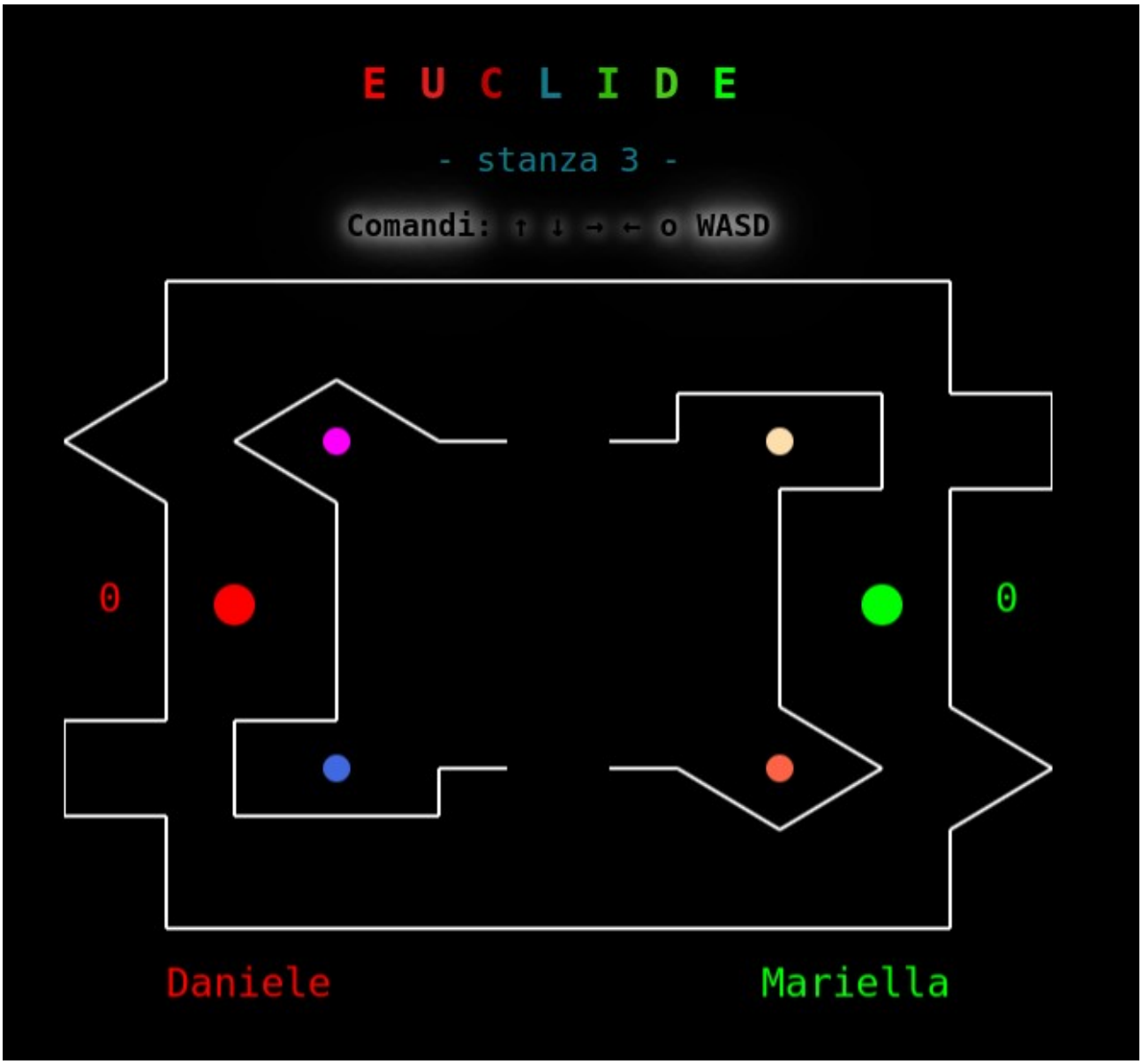
Gli elementi usati per l'interfaccia sono:

1. index.html
2. style.css.

Dopo l'inserimento del nickname e dopo aver premuto su "Start" si presenterà la seconda schermata.

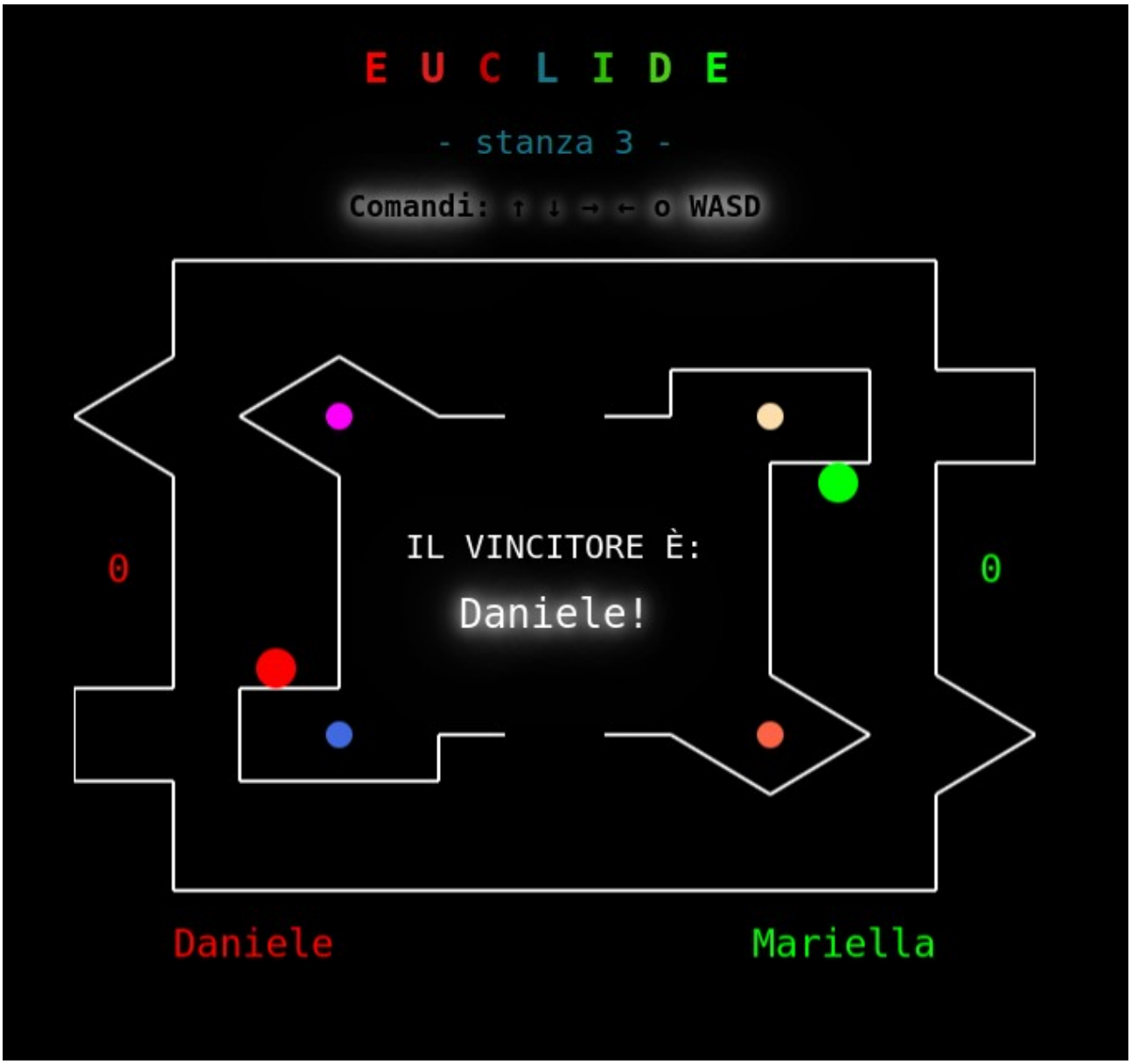
La seconda schermata è stata implementata con l’utilizzo di **Canvas**.

Il Canvas consiste in una regione disegnabile, definita in codice HTML con gli attributi *height* and *width*.
Il codice JavaScript può accedere all'area con un set completo di funzioni per il disegno, permettendo così la generazione dinamica di disegni.



Quando il gioco terminerà, ovvero dopo che uno dei due giocatori avrà realizzato 4 punti, comparirà al centro dell'area di gioco il nome del vincitore.

Sarà data comunque la possibilità di una eventuale rivincita allo sconfitto. Infatti la disconnessione dal gioco avverrà solamente quando l'utente chiuderà la finestra di gioco o la ricaricherà.
Per avere la rivincita quindi basterà solamente continuare a giocare e si rientrerà nella logica di gioco, spingendo le 4 palline piccole che nel frattempo si saranno riposizionate automaticamente nella loro posizione di partenza.



Socket.io rooms

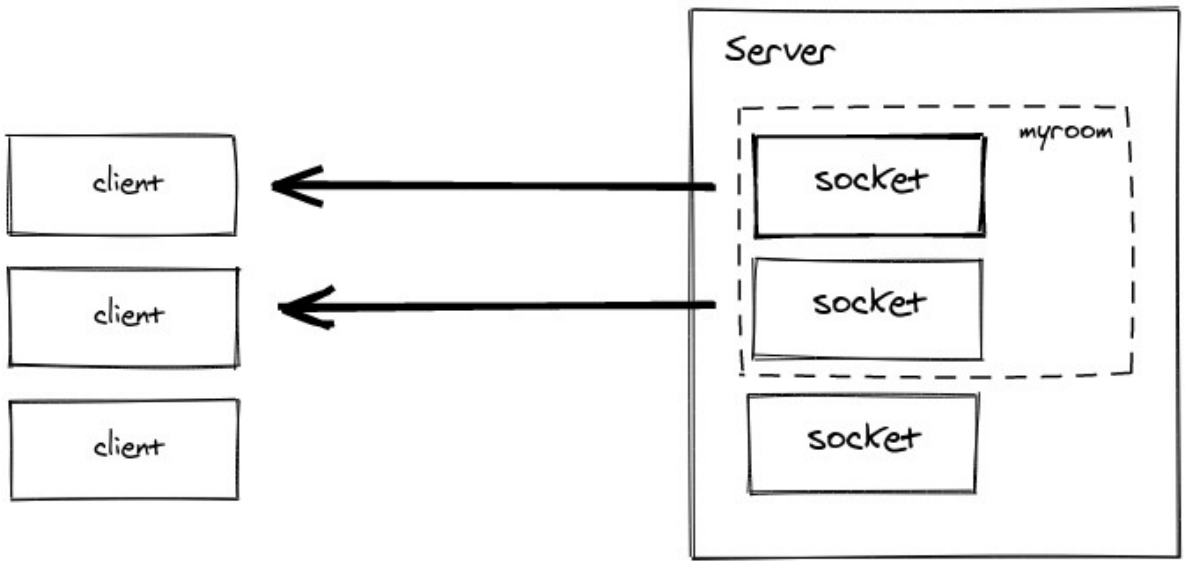
Per avere più di due giocatori che giocano in modo simultaneo in stanze differenti, si è utilizzato socket.io rooms.

Per gestire questa realizzazione, ogni qualvolta un giocatore si connette e trova già la stanza piena, cioè con 2 giocatori all'interno, potrà accedere ad una stanza nuova vuota e aspettare lì per un secondo giocatore.

Il server si occuperà di calcolare la fisica di ogni giocatore in ogni stanza ma invierà solo la posizione delle coordinate e la logica del gioco al giocatore.

I due metodi utilizzati sono join e to:

- Il server può inserire un client in una stanza attraverso la funzione join.
- La funzione to è utilizzata invece per emettere (emit) dal server al client.



Pubblicazione

Per la distribuzione online ho deciso di utilizzare Heroku.

Heroku è una piattaforma cloud di programmazione progettata per aiutare a realizzare e distribuire applicazioni online, una delle più grandi piattaforme PaaS esistenti.

La scelta è ricaduta su Heroku dato che ha ampliato il supporto per il linguaggio di programmazione Node.js. → https://euclidegioco.herokuapp.com/

Su una piattaforma di questo genere è stato possibile distribuire il progetto tra gli utenti sfruttando le risorse del fornitore PaaS.

Infine grazie all'integrazione con Git, è stato così possibile sviluppare il gioco su una piattaforma esterna per poi importare il progetto all'interno dell'account Heroku, lasciando alla piattaforma PaaS il lavoro per renderlo operativo e funzionante.

Siti consultati

- https://socket.io/
- https://nodejs.org/en/
- https://socket.io/get-started/chat → Integrating Socket.IO
- https://expressjs.com/en/starter/hello-world.html
- https://socket.io/docs/v3/rooms/index.html → Stanze.
- https://stackoverflow.com/ → Per utilizzo Canvas e altro.
- https://www.heroku.com/ → Per distribuzione online.

CONCLUSIONI

Con l’implementazione di questo gioco, ho potuto affrontare ed approfondire il problema delle connessioni tra due giocatori.
Lo studio dell’architettura client/server e della gestione reti sono stati fondamentali.

Un futuro sviluppo potrebbe riguardare l’integrazione di un database per memorizzare informazioni riguardante i 2 giocatori come ad esempio i loro punteggi.

Altre idee che ho in mente sono:

- La possibilità di scegliere la difficoltà di gioco.

- Creare altre barriere e vincoli.

- Avere più livelli di gioco.

- Assegnare alle palline valore dei punteggi in modo differente e dunque cambiare il punteggio finale da raggiungere.
  Ad esempio pensavo anche al fatto di assegnare ad alcune palline un punteggio negativo (-1) o (-2), rendendo il gioco più difficile. Ovviamente colorando in maniera significativa e differente quest’ultime, rispetto alle palline con punteggio positivo.

- Dare la possibilità di una maggiore interazione attraverso una chat.

- Cambiare la dinamicità del gioco, quindi aumentando o diminuendo la velocità o l’attrito delle palline.

- Inserire effetti sonori sia per quando viene segnato un punto che in altre occasioni, ad esempio quando la pallina sbatte in una barriera.

- Dare la possibilità di giocare di squadra, per esempio un 2 vs 2 o 3 vs 3.