Progetto FCG – Visualizzatore dati IMU di camminata.

Tommaso Dani

Obbiettivo del progetto:

In questo progetto ho voluto realizzare un software di visualizzazione di dati IMU di una camminata raccolti da sensori.

I dati che restituiscono i sensori comprendono:

1. Accelerazioni sui tre assi del sensore
2. Valore dell'accelerazione G sui tre assi del sensore
3. Velocità angolare sui tre assi
4. Timestamp del dato (quando è stato campionato)

Il dataset analizzato deriva da uno studio dei dati proposto nel corso di FESI.

Idea:

Considerando che dati delle misurazioni erano visualizzabili solo tramite grafici sul tempo, e questi non fornivano un'idea chiara di cosa succedeva nello sviluppo della camminata, ho pensato di creare questo software per permettere una visualizzazione più intuitiva di cosa avviene effettivamente.

Requisiti che mi sono imposto:

1. Devo poter "mouvermi" sui dati e visualizzare un particolare intervallo di dati.
2. Devo poter visualizzare la rappresentazione del complesso gamba/bacino su tre piani ortogonali (laterale, frontale, !laterale).
3. Devo poter dilatare/restringere il tempo sulla riproduzione per poter visualizzare meglio / più rapidamente la rappresentazione dei dati.

Cose in aggiunta:

Ho aggiunto degli elementi che vanno fuori dai requisiti per rendere migliore la grafica e il debug.

Nelle migliorie sulla grafica:

- Ho aggiunto la possibilità di settare la visibilità / trasparenza delle collezioni e dei pezzi.
- Ho aggiunto delle texture per rendere la visualizzazione più rappresentativa.
- Ho impostato un handle per il click con il tasto medio del mouse per fare il drag della vista (paradigma world_in_hand).

Nelle migliorie sul debug:

- Handle per spostamento dei pezzi nella scena (con tasto sinistro del mouse).
- Handle per rotazione dei pezzi.
- Modalità di debug (si può settare in compilazione).

Analisi delle versioni:
Versione v0.1:

Nella versione v0.1 ho caricato su github il progetto che inizialmente stavo sviluppando offline.
In questa versione ho implementato la struttura del software:

- L'organizzazione delle directory
- Costruzione della gerarchia delle classi di visualizzazione, motore della fisica e lettura dei dati

Sono presenti delle categorie di classi come:

- CSV (servono per leggere e caricare i dati CSV dei sensori)
- Pezzi (definiscono i segmenti della gamba e altre strutture della scena)
- Rigidbody (definisce la fisica dei corpi rigidi)
- Joints (definiscono le interazioni e i constraint tra pezzi)

Ho sfruttato l'utilizzo di classi astratte per rendere il codice più semplice da scrivere principalmente per non dover chiamare le funzioni di aggiornamento per ogni tipo di oggetto in modo differente.

Nei pezzi sono presenti:

1. Gamba
2. Caviglia
3. Sensore

Questi concretizzano la classe astratta PieceInterface, sono tutti rigidbody e hanno delle funzioni comuni come update() e draw() per l'aggiornamento della posizione/rotazione nella scena e per disegnare gli elementi.

Nei joint sono presenti:

1. Rigid joint (fissano i movimenti del padre a tutti i figli, rispetto ad un offset di partenza)
2. Pivot joint (fissano i movimenti del padre ad un punto di fissaggio a cui vengono ancorati i figli, che mantengono la capacità di ruotare su quel punto)

In questa versione gli spostamenti dei figli con i joint sono implementati tramite calcoli di trigonomeria, non vengono utilizzate matrici di rotazione.
È presente un main di test e dei file CSV caricati per un primo esempio di rappresentazione di una gamba vista lateralmente.

Per la lettura dei dati da CSV mi sono fatto aiutare da un llm per costruire la classe.

Versione v0.2:

Nella versione v0.2 ho modificato il calcolo dello spostamento dei figli dei joint implementando l'utilizzo di matrici di rotazione.
Per fare questo in modo semplice ho incluso la libreria glm per i calcoli matriciali.

Inoltre ho ridefinito la classe caviglia per poterla utilizzare correttamente.

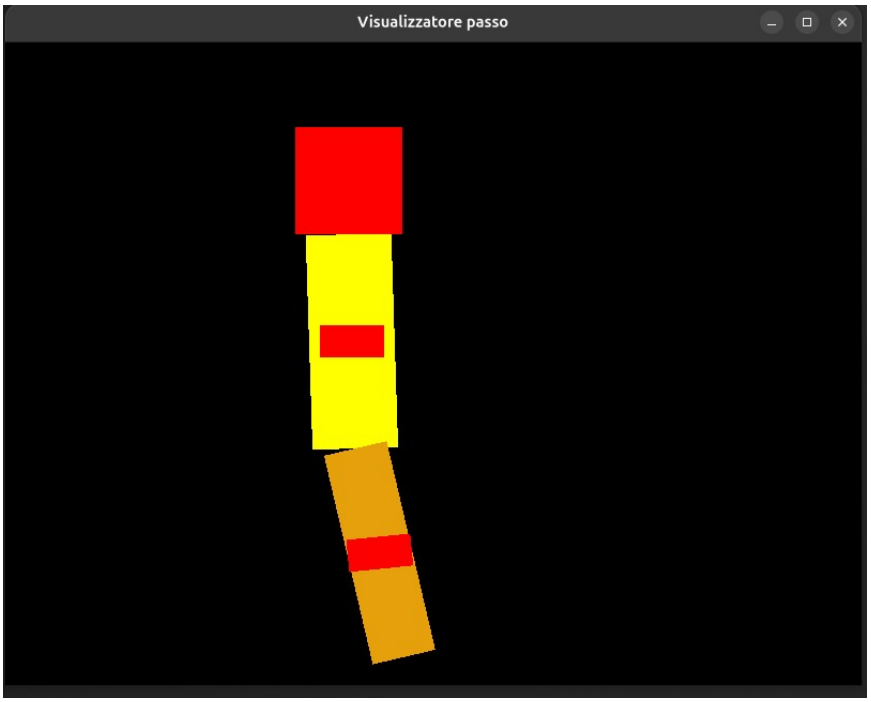
La rappresentazione visiva è rimasta inalterata rispetto alla versione v0.1.
Viene però risolto un bug sul RigidJoint con figli con offset = 0, dove in questa versione rimangono correttamente ancorati al padre mentre in quella precedente potevano muoversi indipendentemente.

Versione v0.3:

In questa versione ho aggiunto un pezzo (classe Torso) per poter ancorare la gamba e definire in modo più solido la struttura.

Inoltre ho applicato refactoring vari alle classi PieceInterface per iniziare ad implementare la vista frontale e altre migliorie del codice.

La rappresentazione è modificata rispetto alla versione precedente e ora la gamba è ancorata ad un rettangolo rosso (il torso).



Versione v0.4:

In questa versione ho aggiunto la possibilità di cambiare piano di visualizzazione.
Per farlo si deve premere la barra spaziatrice.

Per implementare questa vista ho dovuto applicare ulteriore refactoring alle classi PieceInterface per aggiungere una shape da visualizzare su questo nuovo piano.
Ho modificato le funzioni di draw() per implementare il cambio di piano.

Ho dovuto aggiustare i calcoli dei joint per poter spostare i pezzi in uno spazio tridimensionale; inoltre è stato necessario modificare tutti i riferimenti agli assi per definire uno standard all'interno del codice, da qui in avanti:

- Asse X (si trova sempre in testa ai vettori e corrisponde all'asse x di SFML nel piano XZ laterale)
- Asse Y (si trova nel mezzo dei vettori e corrisponde all'asse x di SFML nel piano YZ frontale)
- Asse Z (si trova in coda ai vettori e corrisponde all'asse y negato di SFML, è sempre la verticale)

Versione v0.5:

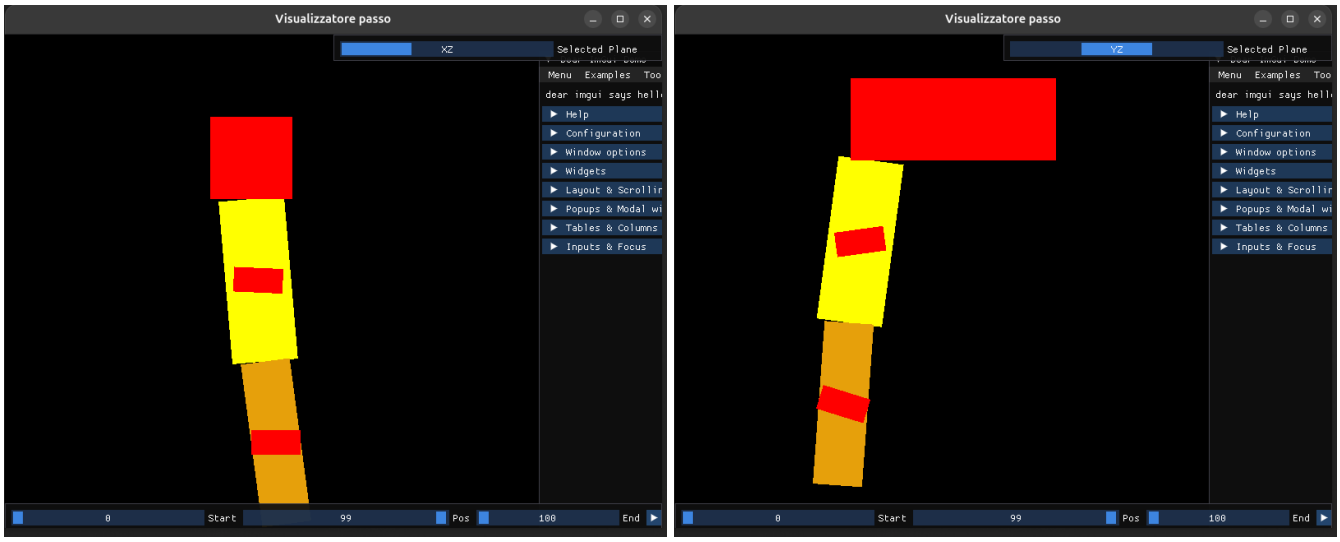
In questa versione ho iniziato a toccare la parte della GUI.
Sono stati implementati 3 slider per il controllo della posizione dei dati in una window di ImGUI:

1. Indice di partenza
2. Indice di visualizzazione
3. Indice di arresto

Ho aggiunto anche un tasto per avviare/interrompere l'animazione.

È stata modificata la classe sensore e il main per aggiungere le variabili di controllo della posizione sui dati.

Ora a differenza della versione precedente in basso è presente la window di controllo.



Versione v0.6:

In questa versione ho aggiunto un'altra categoria di classi: le collezioni.

Le collezioni che ho aggiunto sono:

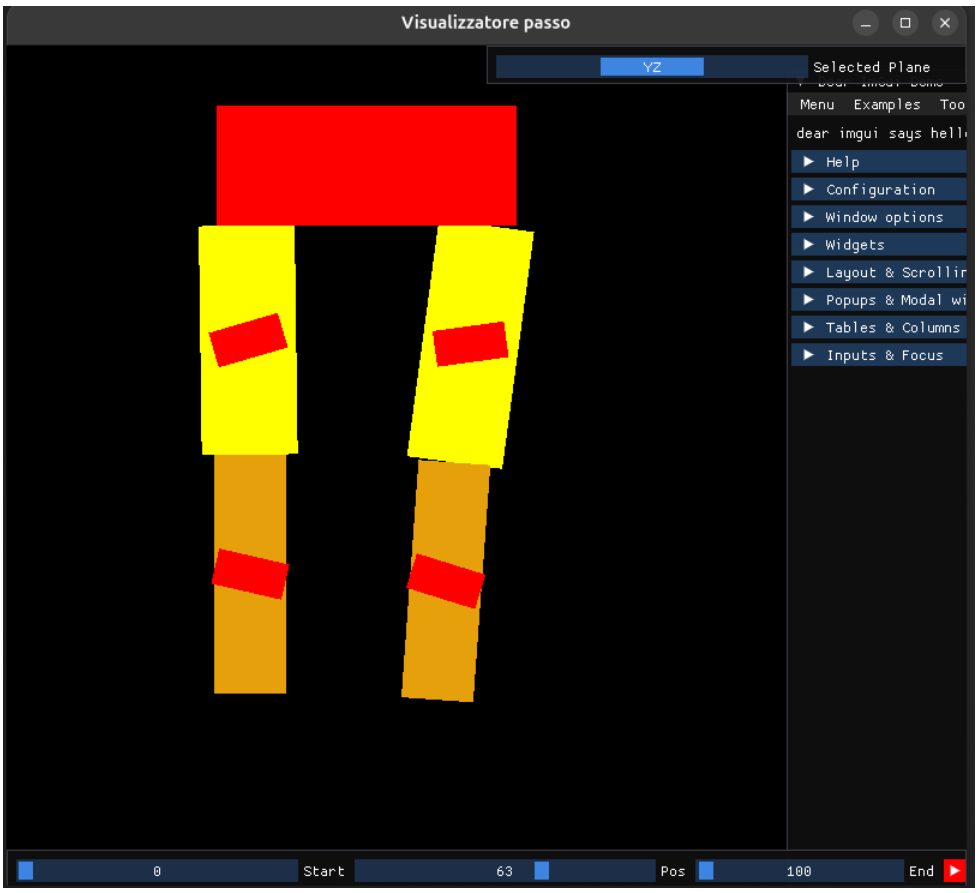
1. Gamba (contiene due pezzi(caviglia e coscia) e informazioni sulla posizione)
2. Lower_body (contiene due Gambe e il torso e definisce il sistema delle gambe)

L'utilizzo delle collezioni semplifica il codice del main perché richiede la costruzione di un singolo oggetto lower_body per definire l'insieme.

Per poter visualizzare le due gambe separatamente ho aggiunto dei file IMU per la seconda gamba.

Inoltre ho aggiunto un tag per il debug che disabilita/abilita lo spostamento dei pezzi tramite mouse; questo si attiva tramite flag in compilazione.

Ora nella rappresentazione si possono vedere entrambe le gambe che si muovono indipendetemente l'una dall'altra, ma il bacino è ancora fissato nello spazio.

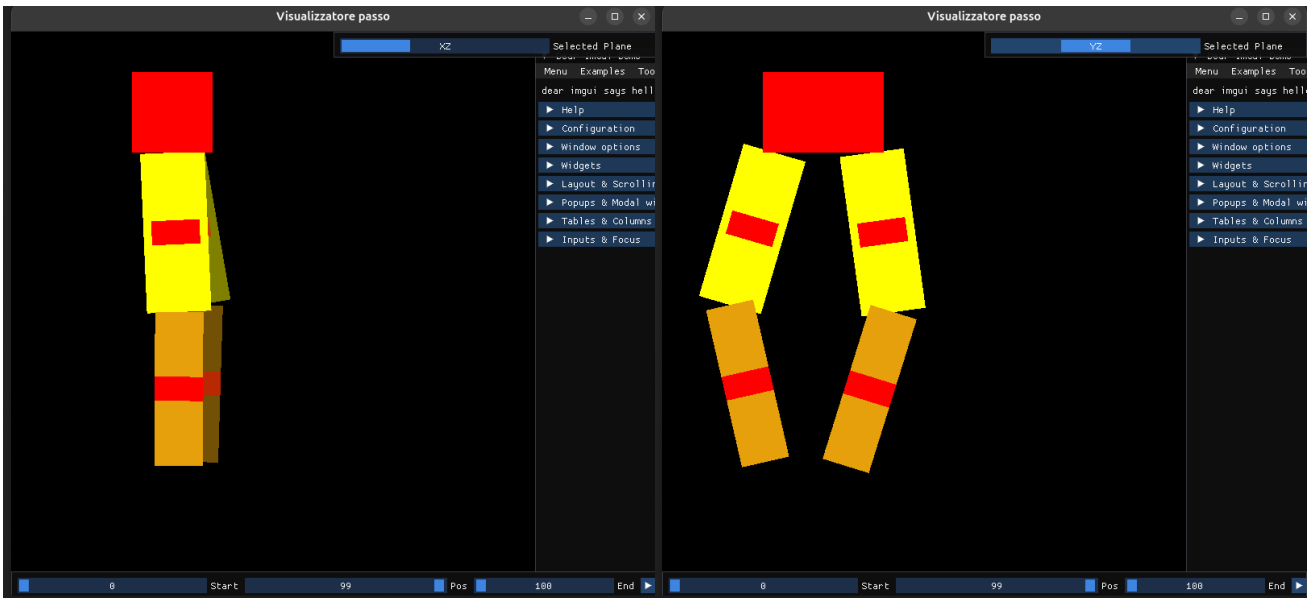


Versione v0.7

In questa versione ho aggiornato le collezioni per implementare il controllo della trasparenza dei pezzi che ne fanno parte.
Questo mi permette di visualizzare la gamba più distante come semi-trasparente con la collezione lower_body.

Ho aggiustato un bug della versione precedente riguardo alla direzione della camminata sul piano laterale, ovvero che la gamba più lontana andava nella direzione opposta.
È stato aggiustato implementando un enum che definisce la direzione e modificando la update() di Sensore.

Inoltre ho aggiustato l'offset di rotazione dei sensori in modo da visualizzare fedelmente i dati derivanti da essi.



Versione v0.8

In questa versione ho implementato l'oscillazione del bacino.
Per fare questo ho Implementato in PieceInterface una funzione che restituisce l'accelerazione verticale del pezzo.
L'accelerazione verticale la ricavo tramite il modulo delle accelerazioni sugli assi del sensore e l'angolo che ha il sensore sull'asse verticale del mondo sottraendo poi il modulo della G (dato che nei dati RAW l'accelerazione di gravità è sommata alle accelerazioni sugli assi).

La collezione gamba restituisce la somma delle accelerazioni di caviglia e coscia.
La collezione lower_body prende i valori delle accelerazioni e calcola la velocità tangenziale del bacino, e dato che ho a disposizione il dt posso ricavare l'angolo spaziato.

Dato che i sensori hanno tanto rumore nei dati ho fatto si che la rotazione del bacino tenda a tornare a quella di partenza in modo tale che questo non continui a ruotare.

Versione v0.9:

In questa versione ho implementato un moltiplicatore della velocità di riproduzione.
Per fare questo ho modificato il codice del main inserendo il moltiplicatore al dt del clock principale.

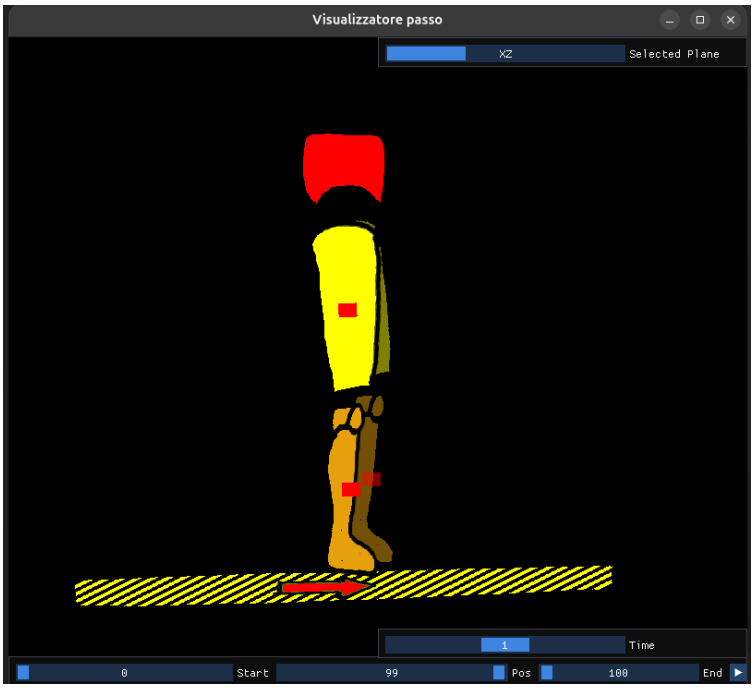
Nella GUI ho aggiunto una finestra con uno slider per impostare il moltiplicatore (può andare da 0.5 a 1.5).

Ho modificato anche la funzione update() dei pezzi per implementare il moltiplicatore.

Versione v1.0:

In questa versione ho cercato di migliorare la grafica generale del progetto.
Ho aggiunto le texture per i pezzi: torso, coscia e caviglia.
Ho aggiustato la direzione delle texture in base al piano di visualizzazione.
Ho aggiunto un nuovo pezzo "pavimento" per definire l'appoggio della gamba e visualizzare la direzione della camminata.

Inoltre ho pulito il main di test per renderlo più facile da modificare anche nel caso di aggiunta di nuovi pezzi/collezioni.



Versione v1.1:

In questa versione ho aggiunto il controllo della visibilità delle collezioni. Ora nella GUI è presente anche una finestra che contiene uno slider e un checkbox per impostare la trasparenza e la visibilità delle collezioni nella scena.

Difficoltà riscontrate:

- ### Creazione e comportamento dei joint
  Ho avuto diverse difficoltà nella creazione dei joint, in particolar modo nel passaggio dal calcolo delle rotazioni da 2D a 3D (principalmente per una questione di sistemi di riferimento). Mi è risultato complesso capire quale fosse l'asse che mi interessava per la rotazione su un determinato piano.
  Ho risolto ridefinendo uno standard sulla direzione degli assi nel progetto.

- ### Movimento/Sovrapposizione dei pezzi in 3D
  Quando ho implementato la visualizzazione sui due piani (laterale e frontale) mi sono imbattuto in un problema di sovrapposizione dei pezzi: siccome i pezzi rimanevano della stessa dimensione e il loro centro di spostamento si trova nel baricentro della shape, quando su un piano un pezzo della gamba ruotava e faceva salire il suo centro, sull'altro piano due pezzi della gamba finivano sovrapposti uno sull'altro.
  Ho risolto facendo una scalatura della shape sull'asse Y di SFML in base al sin dell'angolo di rotazione sulla normale del piano di visualizzazione.

  Un'altro problema era riguardo al drifting sui pivot. Dato che calcolavo la posizione successiva facendo riferimento a quella attuale, questo mi comportava la somma di errori sempre più grandi.
  Ho risolto ricalcolando la posizione dal punto di partenza.

- ### Rotazione del bacino
  Il problema che ho riscontrato era nel calcolo della velocità tangenziale del bacino, ovvero non riuscivo a trovare la velocità corretta e per causa del rumore sui dati mi risultava una velocità sempre più elevata al passare del tempo.
  Ho risolto dopo essermi accorto di aver implementato un bug nella versione v0.5 dove andavo a resettare il clock di base, questo mi causava dei dt negativi e non calcolava correttamente la velocità, ho aggiunto un clock per il motore della fisica.
  Inoltre ho implementato un soft_brake sulla velocità tangenziale in modo da far rallentare la velocità in base all'angolo del bacino.

Utilizzo di risorse:

Mi sono fatto aiutare da un llm per la scrittura di:

- classe CSV (per leggere dati da file)
- file CmakeLists.txt (per implementare la compilazione di diverse build contemporaneamente)
- Consigli sulla struttura delle classi.

Per il resto del codice non ho fatto riferimento a repository esterne, tutto ciò che è nel progetto è stato scritto da zero.

I dati IMU derivano da sensori presi dall'università di Genova e forniti per il corso di FESI.

Le texture sono originali e possono essere cambiate facilmente (fornendo png con la stessa nomenclatura) o rimosse totalmente.

23/06/2026