Politecnico di Bari

Politecnico di Bari

Dipartimento di Ingegneria Elettrica e dell'Informazione

Corso di Laurea Magistrale in Ingegneria dei Sistemi Medicali

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

TEMA D'ANNO IN SISTEMI DIAGNOSTICI TERAPEUTICI RIABILITATIVI AVANZATI

SISTEMI RIABILITATIVI TERAPEUTICI AVANZATI

Professore:
Vitoantonio Bevilacqua

Studentesse:
Benedetta **ALTAMURA**
Martina **CAFERRA**

INDUSTRIAL INFORMATICS LAB
LABORATORIO DI INFORMATICA INDUSTRIALE

Anno Accademico 2022/2023

apulian bioengineering company

INTRODUZIONE

I tumori cerebrali, indipendentemente dalla loro dimensione o posizione, sono in grado di influire in modo significativo sulle capacità fisiche, sociali, professionali ed emotive dei pazienti.

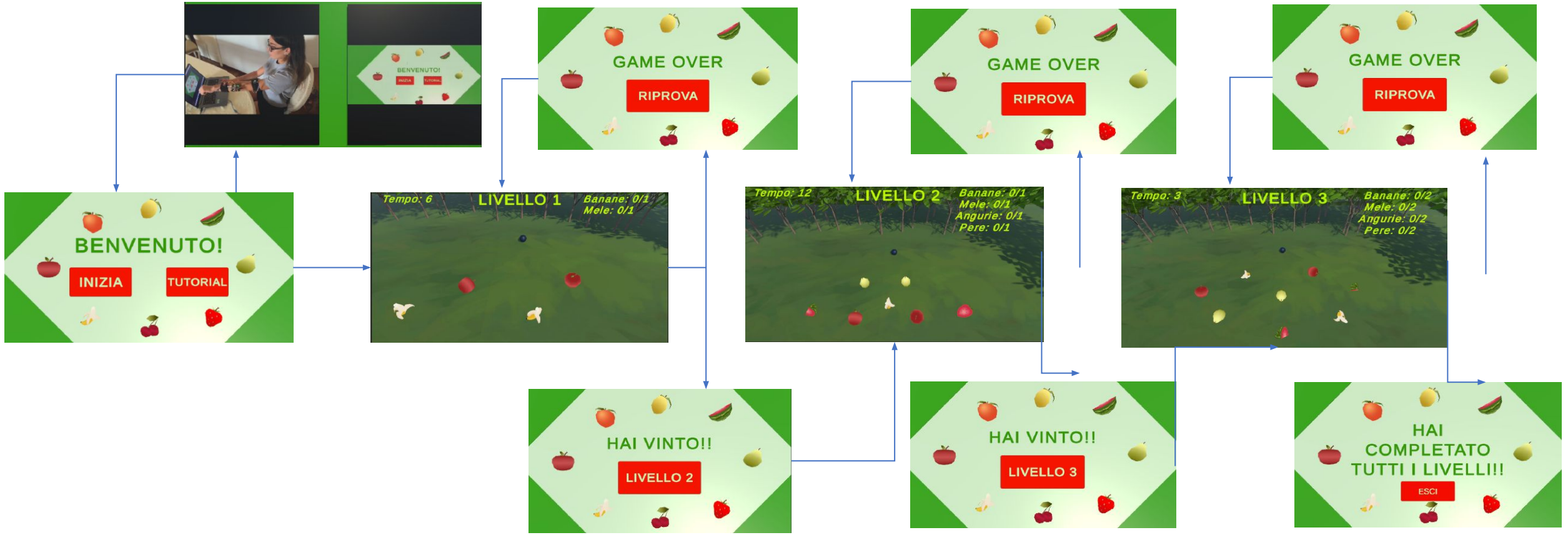
L'approccio riabilitativo deve essere attentamente calibrato per favorire il recupero dell'autonomia individuale, avvalendosi di varie metodologie come la fisioterapia motoria, la terapia del linguaggio e la terapia occupazionale.

Concentrandoci sull'aspetto del recupero motorio, il movimento degli arti colpiti contribuisce a prevenire o attenuare il verificarsi di contratture muscolari e a preservare la mobilità articolare. È altrettanto importante che gli arti colpiti vengano regolarmente esercitati per preservare il tono e la forza muscolare.

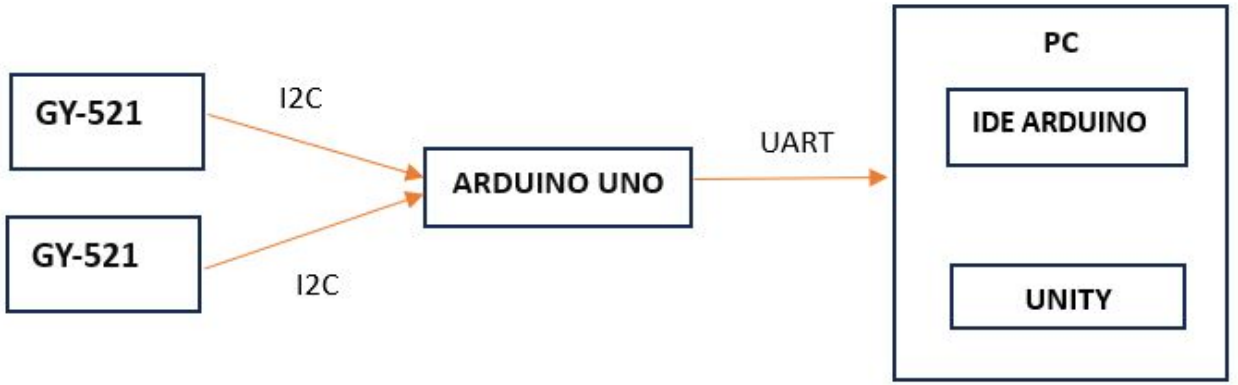
Un'interessante e innovativa strategia nell'ambito della riabilitazione è rappresentata dai "serious game" che possono integrare l'aspetto motorio con quello cognitivo, rendendo l'esperienza di recupero più coinvolgente e motivante per i pazienti. Questi giochi non solo mirano a migliorare la forza, la coordinazione e la mobilità dell'arto, ma integrano anche sfide cognitive che possono coinvolgere l'attenzione, la pianificazione, la memoria e la soluzione di problemi.

STRUTTURA DEL GIOCO

Il gioco prevede che il paziente attraverso il movimento degli arti in un piano, colpisca la frutta richiesta per ciascun livello. Il gioco prevede tre diversi livelli con difficoltà crescente in termini di punti del piano che vengono toccati e in termini di quantità di frutta da collidere



STRUTTURA HARDWARE

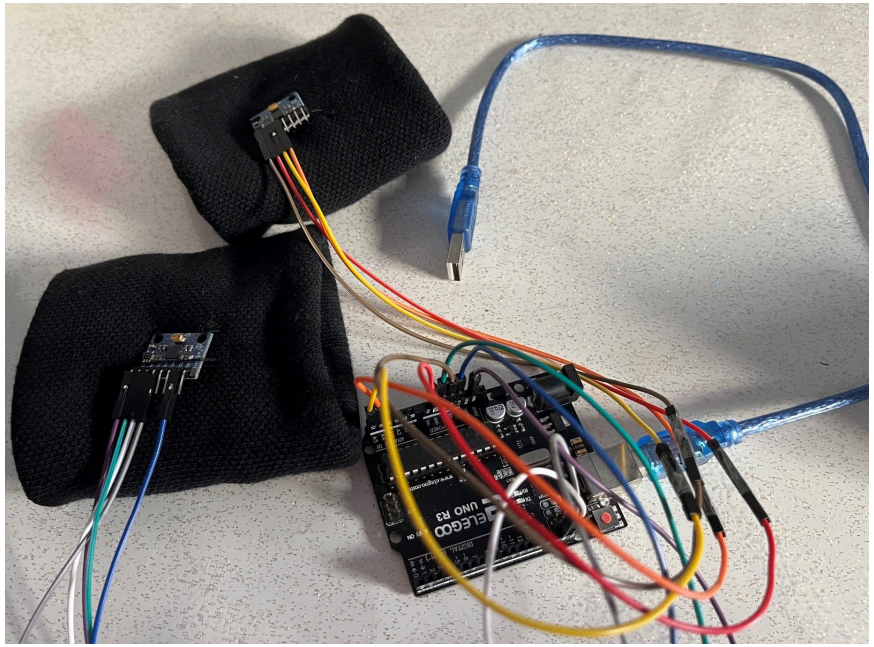


PROTOCOLLI DI COMUNICAZIONE:

1. **I2C** □ protocollo di comunicazione seriale sincrono a due fili che utilizza due linee, SDA per la trasmissione dei dati e SCL per il segnale di clock generato dal dispositivo master.
2. **UART- Comunicazione Seriale tramite USB**□ il protocollo è asincrono, dunque il baud rate usato è 115,2K

MODULI:

1. **GY-521 (MPU-5060)**□ accelerometro e giroscopio triassiali che funzionano attraverso lo spostamento di una massa che modifica la struttura elettrica a cui è collegata. L'output dei sensori in questa applicazione è la velocità angolare
2. **ARDUINO UNO**□ scheda microcontrollore

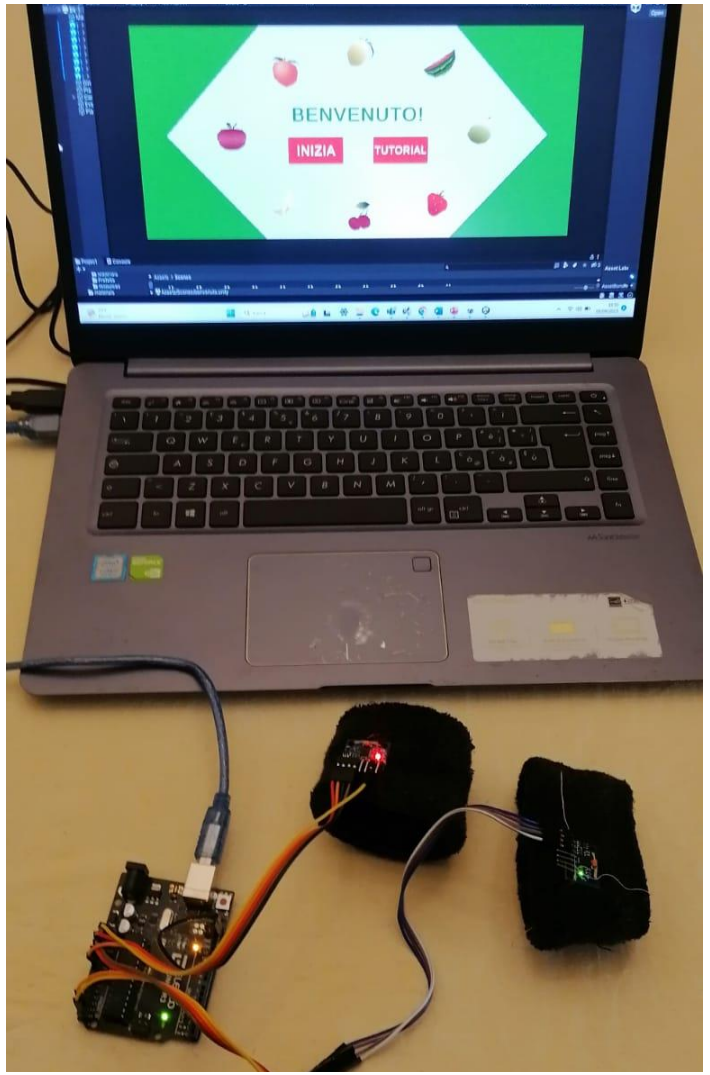


STRUTTURA HARDWARE

ARDITY

Per stabilire la comunicazione seriale tra Unity e Arduino Uno, abbiamo utilizzato un plugin chiamato Ardity, che abbiamo scaricato dall'Unity Asset Store. Questo plugin ci ha permesso di creare un thread (sequenza di istruzioni) dedicato in cui eseguiamo il polling di una porta COM specifica per verificare periodicamente lo stato o i dati disponibili sulla porta seriale (COM port) del computer.

```
- Using System.IO.Ports;
- SerialPort stream = new SerialPort("\\\\.\\COM5", 115200);
- stream.Open();
- stream.Close();
- string UnSplitData = stream.ReadLine();
```



MODELLAZIONE DELLE ARTICOLAZIONI

Vengono acquisiti i dati dei due giroscopi che misurano la velocità angolare su tre assi: X, Y e Z per poi ottenere, integrando, la variazione dell'angolo dalla posizione di partenza lungo i singoli assi dei due MEMS.
Per modellizzare l'arto superiore da un punto di vista ingegneristico, si considerano le articolazioni (spalla e gomito) come dei giunti rotoidali che permettono una rotazione tra due corpi solidi (braccio e avambraccio) su uno specifico asse.

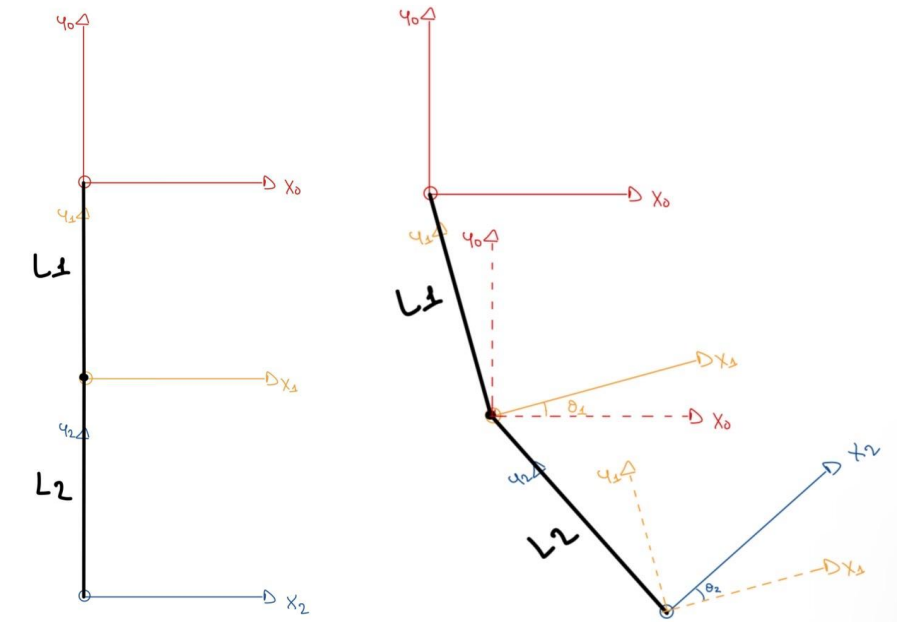
La trasformazione omogenea che descrive la terna $O_1 - X_1Y_1Z_1$ rispetto a quella di riferimento è:

$$A_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & L_1\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & -L_1\cos(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La trasformazione omogenea che descrive la terna $O_2 - X_2Y_2Z_2$ rispetto alla $O_1 - X_1Y_1Z_1$ è:

$$A_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & L_2\sin(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & -L_2\cos(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

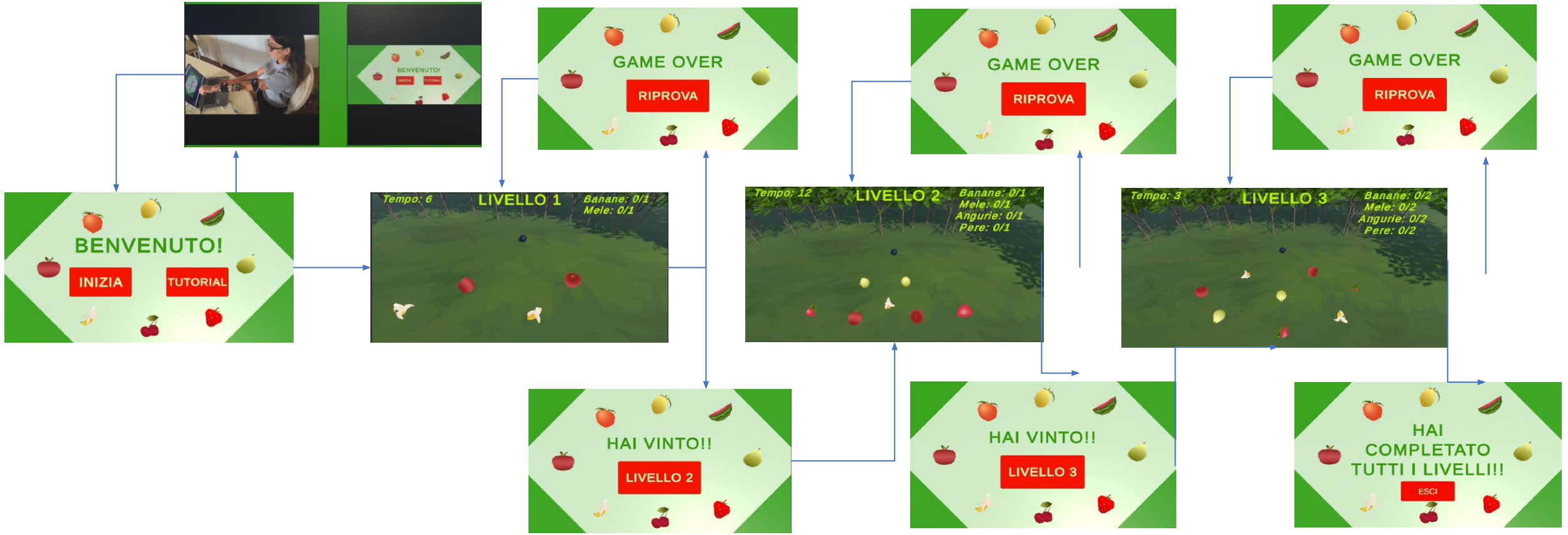
Dunque, la trasformazione omogenea che descrive la terna $O_2 - X_2Y_2Z_2$ rispetto alla terna di riferimento sarà: ${A_2}^0 = {A_1}^0 {A_2}^1$



Conoscendo le coordinate di un punto P nel sistema mobile e l'orientamento del sistema mobile (O'-x'y'z') rispetto al sistema fisso (O-xyz), si può affermare che: p=A*p'.

STRUTTURA DEL GIOCO

Il gioco prevede che il paziente attraverso il movimento degli arti in un piano, colpisca la frutta richiesta per ciascun livello. Il gioco prevede tre diversi livelli con difficoltà crescente in termini di punti del piano che vengono toccati e in termini di quantità di frutta da collidere



COMPONENTI LIVELLI

Il **Player** corrisponde alla sfera che si muove sul piano controllata direttamente dal paziente attraverso i sensori.

```
if (gameOn)
{
    string UnSplitData = stream.ReadLine();          // Legge una linea di dati dalla porta seriale
    string[] dataValues = UnSplitData.Split(',');    // Divide i dati in un array utilizzando la virgola come separatore

    moveHorizontal = float.Parse(dataValues[0], CultureInfo.InvariantCulture);
    moveVertical = float.Parse(dataValues[1], CultureInfo.InvariantCulture);
    UnityEngine.Vector3 positions = new UnityEngine.Vector3(-moveHorizontal, 0, -moveVertical - 10);
    UnityEngine.Vector3 direction = positions - transform.position;
    direction.Normalize();
    UnityEngine.Vector3 velocity = direction / Time.deltaTime;
    rb.MovePosition(positions);
```

- Mesh filter
- Mesh render
- Sphere Collider
- Rigid body
- C# script: posizione, eventi di collisione, aggiornamento UI, stato del gioco

La **frutta**, scaricata dall'Unity Asset Store gratuitamente, rappresenta l'oggetto da collidere.

Ogni tipologia di frutta ha un Tag coincidente con il nome della stessa.

- Mesh filter
- Mesh render
- Sphere Collider
- Tag coincidente con il nome del tipo di frutta.
- C# script rotazione.

```
if (gameOn && other.gameObject.CompareTag("Banana"))
{
    other.gameObject.SetActive(false);  //quando collide scompare

    score_banana++;
    UpdateScore_banana();

    if (score_banana == 1 && score_apple == 1)
    {

        gameOn = false;
        gameWin = true;
    }
    else if (score_banana >= 2 && score_apple >= 0)
    {

        gameOn = false;
        gameWin = false;
    }
    else if (score_banana >= 0 && score_apple >= 2)
    {

        gameOn = false;
        gameWin = false;
    }
```

COMPONENTI LIVELLI

Canvas □ funge da contenitore per tutti gli elementi UI, come testo, immagini, bottoni e pannelli, e fornisce un modo per posizionarli e organizzarli all'interno della scena di gioco. Ad esempio, in ogni livello viene visualizzato il tempo di gioco, il numero del livello attuale e il punteggio associato alle collisioni tra il giocatore e gli oggetti da raccogliere.

Il **ground** e **walls** □ delineano la zona entro cui si posizionano i vari elementi della scena e sono caratterizzati da: Mesh Filter (rispettivamente plane e cube), Mesh Render e Collider (rispettivamente mesh e box).

Main Camera □ seguirà la scena spostandosi con il player grazie a un C# script (CameraController).

```
void Start()
{
    offset = transform.position - player.transform.position;
}

// Update is called once per frame
Messaggio Unity | 0 riferimenti
void Update()
{
    //transform.position = offset;
    transform.position = player.transform.position + offset;

}
```

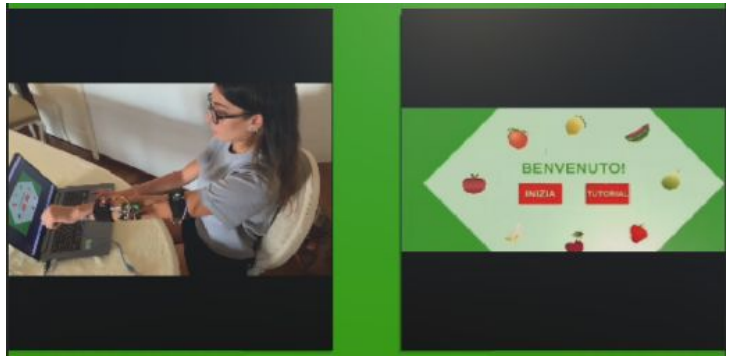
Viene inizializzata la variabile offset calcolando la differenza tra la posizione iniziale della telecamera e la posizione iniziale del giocatore in modo da determinare la distanza iniziale tra la telecamera e il giocatore. Ad ogni frame, la posizione della telecamera viene aggiornata in modo che segua il giocatore.

ALTRE SCENE

- SCENE DI VITTORIA
- SCENA DI GAME OVER
- SCENA BENVENUTO
- SCENA COMPLETAMENTO DEI LIVELLI

Ad ogni **Button** è associato uno script per il caricamento della scena o per uscire dal gioco

- SCENA TUTORIAL le componenti **Video Player** sono associate a dei **Quad** tramite dei **Render Texture**



FUNZIONAMENTO

SCRITTURA DEI FILE XLS E VISUALIZZAZIONE DELLA TRAIETTOIA

Utilizzando la classe "StreamWriter", generiamo due file di output in formato xls (Excel) in cui registriamo la data (giorno/mese/anno) di ogni sessione di esercizi:

```
PosFilename = DateTime.Now.Day + "_" + DateTime.Now.Month + "_" + DateTime.Now.Year + "_LogPosition.xls";
LivFilename = DateTime.Now.Day + "_" + DateTime.Now.Month + "_" + DateTime.Now.Year + "_LogLivelloTempo.xls";
```

Entrambi i file vengono aperti in modalità append:

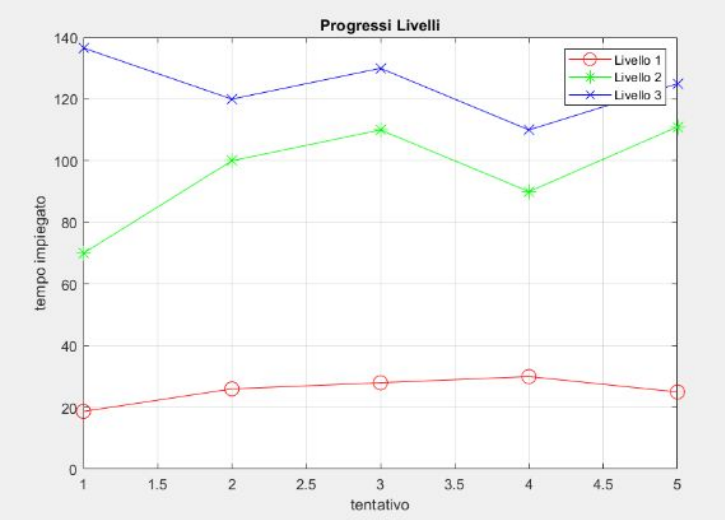
posLogger = new StreamWriter(Path.Combine(pathSave, PosFilename), append: true);

e con il metodo “WriteLine” scriviamo le informazioni di interesse per ciascuno dei due file cioè:

```
//scrivo intestazione di ogni colonna del file
posLogger.WriteLine("LIVELLO"+ "\t"+ "TEMPO" + "\t" + "X" + "\t" + "Y");
livLogger.WriteLine("LIVELLO" + "\t" + "TEMPO TOTALE");
```

nel primo abbiamo le posizioni assunte dal player istante per istante, nel secondo il livello e il tempo complessivo per svolgerlo.

SCRITTURA DEI FILE XLS E VISUALIZZAZIONE DELLA TRAIETTOIA

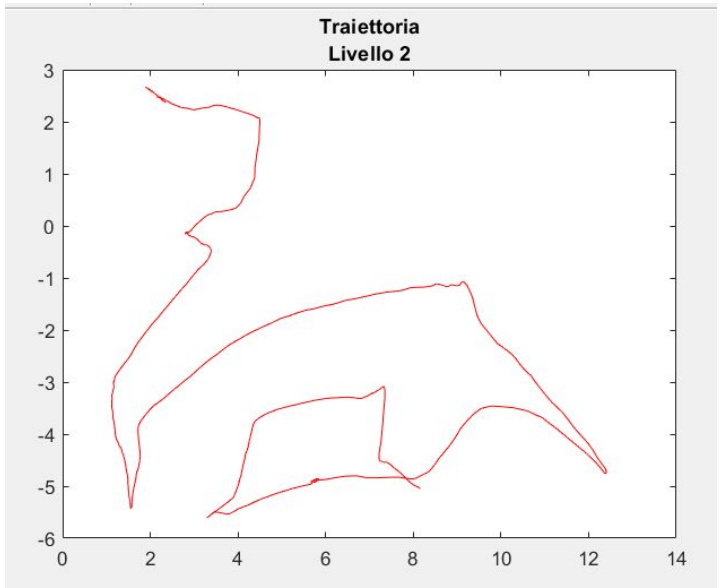


È stato realizzato in matlab un plot che rende visibile:

- Sull'asse x: il tempo impiegato dal player per completare ciascun livello
- Sull'asse y: il numero di tentativi effettuati.

In caso di più tentativi dello stesso livello abbiamo fatto la media dei tempi.

Questi valori vengono plottati grazie ai file excel in cui vengono salvate le informazioni da Unity.



È stato realizzato in matlab un ulteriore plot che rappresenta un esempio di traiettoria:

- Sull'asse x: la variazione di posizione lungo l'asse x
- Sull'asse y: la variazione di posizione lungo l'asse y

GRAZIE PER L'ATTENZIONE