A Multi-agent Body Tracking Application Framework Applied to Physical and Neurofunctional Rehabilitation

Felipe Reis Valente[1(✉)], Marcelo de Paiva Guimarães[2,3], Elder José Reioli Cirilo[1], and Diego Roberto Colombo Dias[1]

[1] Universidade Federal de São João del-Rei, São João del-Rei, Minas Gerais, Brazil
felipevalente@ufsj.edu.br
[2] Universidade Federal de São Paulo, São Paulo, Brazil
[3] Centro Universitário Campo Limpo Paulista, Campo Limpo Paulista, Brazil

Abstract. Stroke is one of the most disabling diseases today, so investigating methods for rehabilitation of post-stroke patients is of utmost importance. Thus, significant benefits can be achieved by using body tracking systems and virtual environments. However, the development of such applications involves a large set of requirements, such as the construction of virtual environments, the interaction devices, and the storage and processing of data collected during rehabilitation sessions. This paper presents a multi-agent framework that abstracts the difficulties involved in developing applications using body tracking devices and virtual environments in neuromotor and neurofunctional rehabilitation.

Keywords: Body tracking · Framework · Multi-agent and rehabilitation

1 Introduction

Events such as stroke and degenerative diseases limit mobility, as does advancing age or accidents that a person may have suffered [36]. Consequently, people who have suffered a stroke need rehabilitation, which is a process of therapy that aims to partially or entirely recover a patient's motor abilities. However, traditional rehabilitation with traditional aids (e.g., walkers and crutches) do not allow an objective analysis of the evolution of the patient's condition [8].

Among the difficulties that physical therapists encounter in the process of rehabilitation of patients, there are those related to the metrics because they are not easy to be observed visually, often being subjective to evaluate the patient's progress [31]. Accurate metrics require rehabilitation equipment with sensors that monitor movements more accurately, providing more detailed feedback to the physical therapist. The physical therapy process can also be enhanced with the use of games allied to motion sensors or depth cameras, as they can facilitate

Supported by organization UFSJ.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13382, pp. 459–472, 2022.
https://doi.org/10.1007/978-3-031-10592-0_33

the execution of new functional motor and or cognitive tasks [20]. Thus, it is possible to combine technology with health care using serious games, which are games designed to achieve a relevant goal, i.e., that go beyond the idea of entertainment, offering learning and training possibilities [3].

The use of depth cameras plays a central role in capturing human body movements for body tracking applications. Through infrared sensors, joint data is captured and can be stored for later playback [23]. There are also inertial sensors (e.g., accelerometers, magnetometers, and gyroscopes). They can infer the position and orientation of tracked objects [32]. Based on the data collected from the patient, the physical therapist can verify that movements are being performed appropriately [9].

This paper presents a framework that facilitates the employment of motion sensors by encapsulating the functionality of different interfaces, either wired or wireless. The developed framework allows developers to quickly develop new applications, increase productivity, reduce error-prone code, improve maintenance, and standardize the project. A case study is also presented in which a new application is developed.

The rest of the paper is organized as follows: Sect. 2 presents the theoretical framework used to build the framework, such as design patterns and motion sensors; Sect. 3 presents our framework, describing its classes and methods. Finally, Sect. 4 presents the conclusions of the work.

2 Theoretical Referential

2.1 Framework

According to Gamma [16], a framework is defined as a set of classes that cooperate, providing a reusable design for a specific domain of system classes. Their use by software engineering provides the developer with a context-specific modular structure, allowing the reuse of components and the coupling of new technologies, creating a new application [4].

The literature shows several frameworks focused on the context of motion capture. Amiri [2] presents a framework for assistive technology-based mobility capture and tracking in individuals with spinal cord injury. Some authors explore the use of Kinect as a body-tracking device [7,26,27,33,34]. [6] aims to create a framework for coupling serious games into a robot that provides corrective feedback during user interaction. [22] shows a framework for monitoring and improving the quality of physical therapy exercises. An Android smartphone application connects to the wristband, transmits the data in real-time for storage in the database, and shows the movement's evolution.

The work of [11] created a framework using Unity to analyze the effectiveness of using a 3D camera to stabilize the reading of data from inertial sensors in the context of rehabilitation. The work of [28] is focused on human activity recognition, for example, identifying whether a person is sitting, walking, or standing, among others. [18] presented a framework that integrates an omnidirectional treadmill with virtual reality to provide a unique rehabilitation experience.

The objective of most of the works is to create a framework for motor rehabilitation using Kinect. Some works have different proposals, such as: creating a framework for noise reduction to improve the accuracy of captured movements, comparing performance in physical activities, using virtual and augmented reality with Kinect, and creating a framework for wearable clothing.

2.2 Neurorehabilitation

Neurorehabilitation is a form of non-invasive intervention to treat cognitive impairment without surgical or pharmacological intervention. It is also valid for patients with memory problems due to Alzheimer's disease. Stroke is one of the most significant causes of neuromotor debilitation. Approximately 80% of the patients who survive the most severe phase are left with sequels afterward. Most can regain the ability to walk; however, 30% to 66% cannot use their upper limbs after suffering a stroke.

Neurorehabilitation treatments are possible because the Central Nervous System is a flexible structure in which its cells can be regenerated, and through specific physiotherapy, the patient can recover lost movements. The Central Nervous System has great adaptability even in adult brains [29];

The application of body-tracking games in rehabilitation centers to aid patient therapy is a growing trend because, with the cheapening of the technology, it is even easier to bring the equipment to the patient's home. In this way, it is possible for the patient to have physiotherapy at home while the data is captured and sent to the physiotherapist remotely to be analyzed later. That makes it easy to use in remote and hard-to-reach places, such as rural areas, requiring only the internet to send the data.

2.3 Body Tracking

The human body is composed of the skeletal system and the articular system. The bones are connected through joints, with the joints acting as axes and the bones as levers, thus favoring the mobility of the human body [21];

The body tracking is responsible for capturing each joint's X, Y, and Z positions. In our framework, 20 joints are monitored at a frequency 30 Hz.

The human body can be divided into three planes: Coronal, Transverse, and Sagittal. The coronal plane passes over the coronal suture of the skull, dividing the body into posterior and anterior; the transverse plane divides the body into two parts, one closer to the ground as inferior and one farther away as superior; and the sagittal plane defines as the plane that passes over the sagittal suture of the skull, dividing the body into left and right [10].

Optical Tracking. Computer vision is an area of research that aims to simulate human vision. The machine receives as input an image, making the extraction, characterization, and interpretation of information from the real world through the captured image [19]. The machine can do body tracking due to it being able

to process and interpret the images captured from the physiotherapy sessions [14].

Optical Tracking allows the user to be tracked without the need for any device attached to the body, i.e., without any wires or transmitters, using only a video camera, which offers good freedom to the user. It also allows the capture of multiple users. The capture areas can be larger than other tracking systems. Although there are several advantages to using Optical Tracking, there are also some disadvantages, such as pre-processing, which requires more computing power, occlusion problems, and the ambient lighting that needs to be controlled.

Inertial Tracking. Inertial Tracking uses measurement sensors attached to the user's body, such as accelerometers, magnetometers, and gyroscopes. With the data captured by these sensors, the angular velocity and linear acceleration are obtained, allowing the position and orientation of the tracked objects to be calculated.

Due to the advancement in technology, inertial devices have become smaller and cheaper, which has made it possible to use these devices for body tracking. However, miniaturization also brought problems such as decreased accuracy. Unlike Optical Tracking, it does not suffer from occlusion problems but can suffer from electromagnetic interference.

2.4 Biomechanics Sensor Node

The Biomechanics Sensor Node (BSN) device was created to miniaturize the body tracking device based on inertial sensors, as it is a device that can be attached to various parts of the user's body. In addition, it has a battery and good ergonomics for use. The communication of the device is performed by using Bluetooth Low Energy (BLE). The BSN has several sensors, such as a gyroscope, accelerometer, and compass, which in combination allow for body tracking of the user [5].

2.5 MediaPipe

MediaPipe is an open-source framework developed by Google, meant to be used on mobile devices and desktops with the purpose of facilitating development when working with user gesture detection.

It is also possible to use MediaPipe combined with machine learning techniques to detect and track objects, faces, and even strands of hair from a person [1]. Machine learning techniques achieve good real-time results even on more limited devices, such as mobile devices. MediaPipe uses two approaches: first, it delimits a box around the object in a complete image, identifying the object to be tracked, and once the object is found, the second approach works only on the delimited region to identify specific key points.

2.6 Kinect

Kinect is a body-tracking device developed and maintained by Microsoft. It is capable of capturing all of the user's joints, as well as their voice. The great thing about this device is that it allows optical tracking through structured light (infrared). Through the sensor, it is possible to reconstruct the three-dimensional space with millimeter precision, capturing the user's depth between 0.8 m and 3.5 m.

Initially, Kinect was developed to be used in electronic games on the Xbox360 console, but with the popularization of the device, today it is being used in Medicine, Robotics, Physiotherapy, Physical Education, among others [12,24,25,35]. Kinect has good documentation and a highly regarded Software Development Kit (SDK) that facilitates the development of the most diverse applications.

3 Our Framework

In the framework developed in this work, we used the following design patterns: (1) the Abstract Factory, which provides an interface for creating families of related or dependent objects without specifying their concrete classes [17]; (2) the Repository, which is a way to encapsulate details of the data access infrastructure, mediating between the data mapping layers and the domain layers [15]; (3) and the Strategy which defines a family of algorithms and arranges the algorithms into separate classes, making them interchangeable [13]. We employed the multi-agent concept in the framework to abstract the difficulties in developing solutions that involve body tracking devices on different platforms. We developed the framework as a Unity asset.

The main class of our framework is the **Core**, whose purpose is to synchronize all the others, which are the following: (1) **Input**; (2) **DataManager**; (3) **Analysis**; and (4) **Utilities**. It is necessary to import its package into Unity to use the developed framework. Thus, it is possible to use the classes already defined with their respective methods, allowing the developer to focus exclusively on the application, not worrying about the particularities of the body tracking devices, database, authentication, data import, and machine learning approaches.

3.1 Input Module

The project pattern used in the **Input** module was the **Abstract Factory**, which is of the creative type since it allows you to create families of related objects without having to specify their concrete classes. It provides some advantages like knowing that the products from a factory are compatible, easy to maintain, and avoid coupling between concrete products and client code. As a disadvantage, it can make the code more complex at first because of the larger number of classes and interfaces [30]. The class diagram shown in Fig. 1 represents the module **Input**.

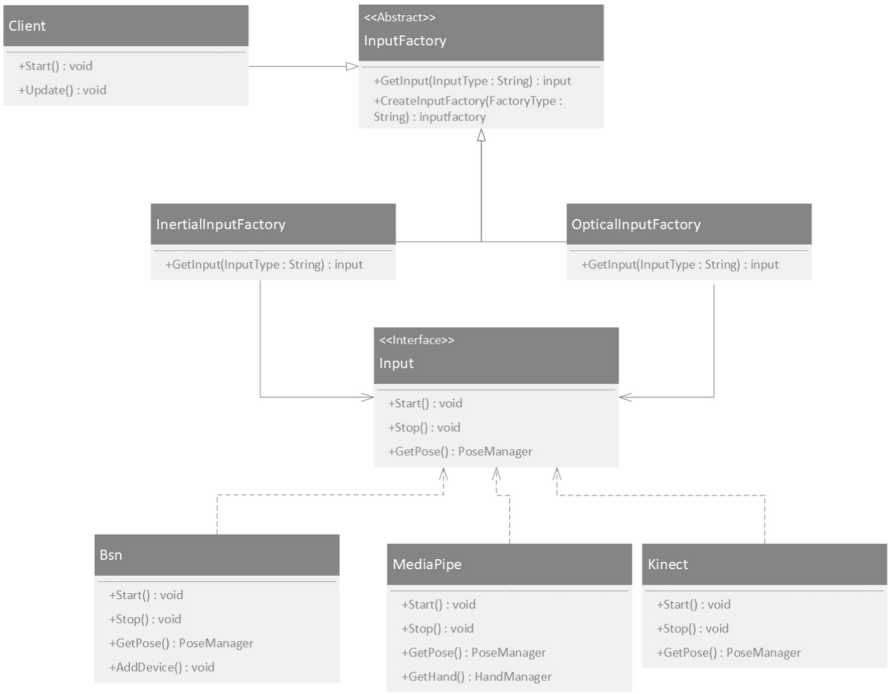

Fig. 1. Class diagram of the **Input** module

In the **framework** the **AbstractFactory** declares an interface for operations that will create **AbstractProduct** objects. In the **framework** these are input devices. This class contains two methods. The **GetInput** method, which is an abstract method that will be implemented by the child factory classes; and the **CreateInputFactory** method, which receives an input parameter (factory type) and then creates and returns the appropriate factory object to the caller.

The concrete class implements the abstract class **AbstractFactory (InputFactory)**. In our work, this class implements the **GetInput** method of the **InputFactory** class. There are two concrete types of factories in our example: **InertialInputFactory**, for inertial devices; and **OpticalInputFactory**, for optical devices.

The method **GetInput** creates and returns the appropriate inertial object based on the input parameter, that is, the received **InputType**. Analogously, the same is true for the optical devices of the concrete class **OpticalInputFactory**.

An interface is used to declare the methods of the Product. In our framework, these are the **start**, **stop** and **getPose** methods of the **Input** object.

The **start** method is responsible for starting the device; the **stop** for stopping the started device; and the **getPose** for returning the registered joints. The concrete product class implements the **AbstractFactory (InputFactory)** interface to create concrete products. In our example, the product classes are:

MediaPipe, BSN and Kinect. The BSN product class has one more method, **AddDevice()**, responsible for pairing new Bluetooth wristbands, thus increasing the number of tracking points.

3.2 DataManager Module

The **DataManager** module is responsible for handling the data captured via the devices, from an external file to the application, or simply using the captured data directly in the application and persisting it to disk. The class diagram shown in Fig. 2 represents the **DataManager** module.

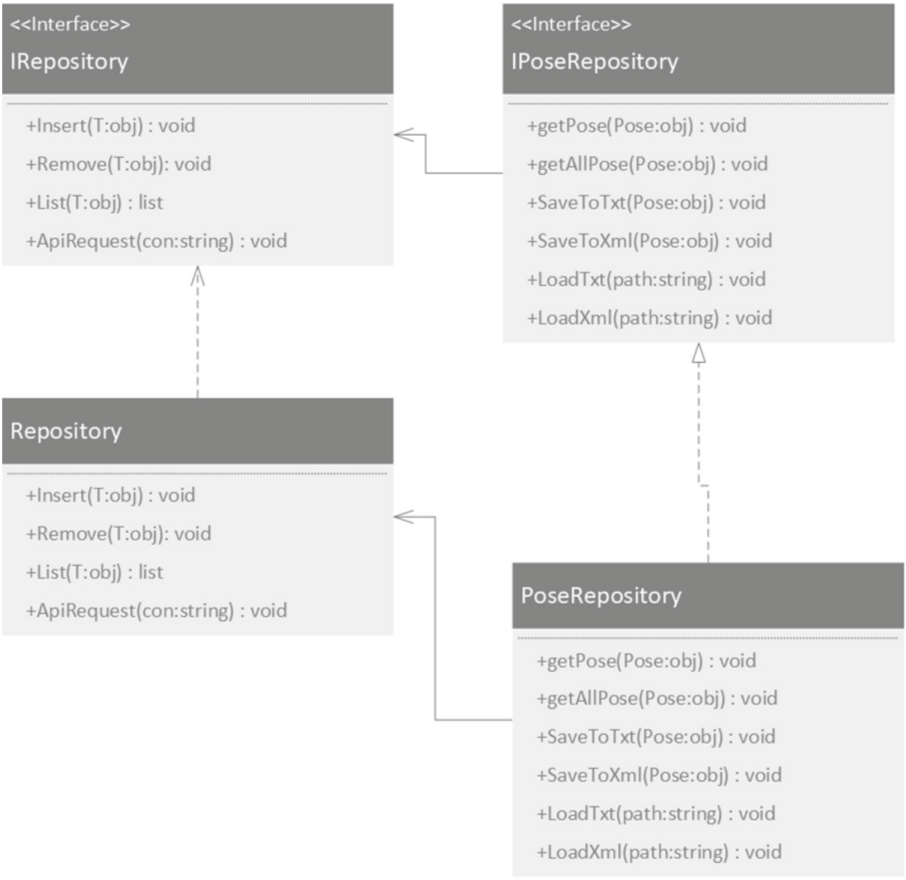

Fig. 2. Class diagram of the **DataManager** module

We employed the **Repository** design pattern to create an abstraction layer between the data access layer and the business layer of an application. Advantages of using this pattern are easier testing, simpler application management

with the data store, and easy switching between multiple data stores without changing the application programming interface (API).

The **IRepository** interface is a generic interface that defines some methods, such as **Insert**, responsible for inserting into the DBMS; **Remove**, for deleting; **List**, for searching; and **ApiRequest**, for working with the API.

The **Repository** is a generic class that implements the **IRepository** interface. Note that the generic repository does not have specific functions to work with the joints.

The **IPoseRepository** interface is a specific interface that defines some specific methods to work with the articulations, such as **getPose**, responsible for returning a specific articulation; **getAllPose**, which returns all articulations; **SaveToTxt**, which persists on disk in a txt file; and **SaveToXml**, which persists on disk in an XML file. The **LoadTxt** and **LoadXml** load joints from a text and XML file, respectively.

The **PoseRepository** is a specific repository that implements the **IPoseRepository** interface, with its methods for handling joints, unlike the **Repository**, which is a generic repository.

3.3 Analysis Module

The **Analysis** class is responsible for analyzing the data; being possible to use machine learning models to identify the type of movement done and score the movement by comparing it with the historical base to follow the patient's evolution, among other possibilities. The class diagram representing the Analysys module is shown in Fig. 3.

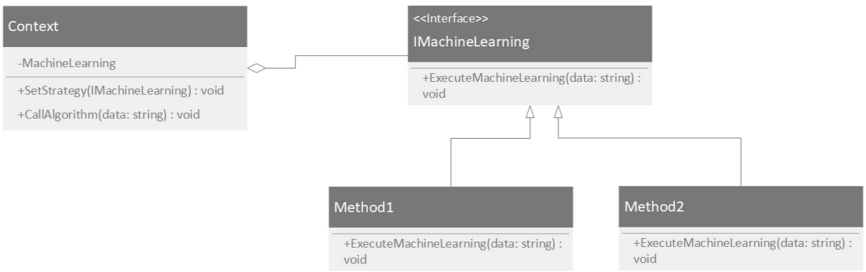

Fig. 3. Class diagram of the **Analysis** module

We employed the behavioral design pattern **Strategy** in the development of this module because several related classes differ only in their behaviors, and variants of the machine learning algorithms [13] are required.

The **IMachineLearning** class declares an interface that is common to all machine learning algorithms. That declares a method that the application context uses to execute some approach.

Each concrete class must implement the **ExecuteMachineLearning** method of the **IMachineLearning** interface. If the need for more algorithms arises, the developer needs to add new concrete classes that implement the interface mentioned above. The **ExecuteMachineLearning** method receives a string with data to be processed. Thus, the concrete class implements different variations of an algorithm that the application context requires.

The **Context** class contains a property that is used to contain the reference of a **strategy** object. The client sets this property according to the required machine learning algorithm.

The **Client** class creates a specific **strategy** object and passes it to the context. The context defines a setter that allows the client to change the strategy associated with the context during execution.

3.4 Utils Module

The **Utils** class is a supporting class to facilitate the developer to work with the data obtained from body tracking, as per Fig. 4.

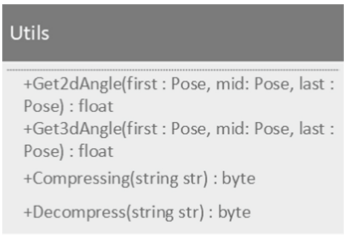

Fig. 4. Utils class structure

We implemented the **Get2dAngle** method to get the angle when working in two dimensions. Three positions must be entered: start, middle, and end. The **Get3dAngle** method works analogously, however, in three dimensions. The **Compressing** and **Decompress** methods compress and decompress data, making it easier to manipulate data when working with databases.

The four defined modules: **Input**; **DataManager**; **Analysis** and **Utilities** define the structure of the framework. The synchronism of the **Core** class with the **Input** and **DataManager** modules allows the framework to be compatible with devices from different platforms, for example, allowing Kinect data, which only works on Windows operating system, to be used on Android operating system.

4 Using Our Framework

We created a simple example as a Unity project with an avatar moving with the data captured by MediaPipe and Kinect. With a new project created, the

avatar in the scene, and the framework imported, we used the Pose Manager component to deal with 20 avatar articulations. We placed this component script in the avatar (Fig. 5).

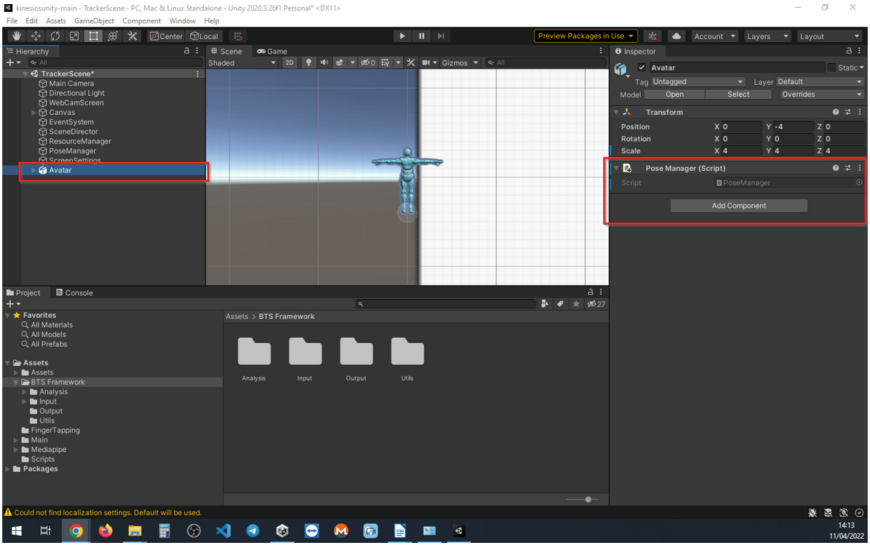

Fig. 5. Pose Manager component

All the libraries needed to work with Kinect and MediaPipe are already included in the framework, so all the compilation and configuration were unnecessary for the devices to work.

The code responsible for working with the Input must be included in the project in the start method. The Client uses the *AbstractFactory* and *AbstractProduct* interfaces to create a family of related objects. We will use MediaPipe capture (Code 1.1).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

namespace AbstractFactoryDesignPattern
{
    class Client: MonoBehaviour
    {
        // Start is called before the first frame update
        void Start()
        {
            Input input = null;
            InputFactory inputFactory = null;
```

```
            // Create the OpticalInput factory object by
                passing the factory type as Optical
            inputFactory =
                InputFactory.CreateInputFactory("Optical");

            // Get MediaPipe Input object by passing the
                input type as MediaPipe
            input = inputFactory.GetInput("MediaPipe");
        }

}
```

Listing 1.1. Client

If the developer wants to use a Kinect, it is effortless if they are using our framework. They just need to change **GetInput("MediaPipe")** to **GetInput("Kinect")**. Pose Manager works with any device without any need to map the joints for each device.

It is necessary to add the code **input.Start()** to start the device and start capturing the joints. If the developer wants to stop the device, they must use **input.Stop()**. With these steps, it is now possible to move the avatar with the data captured by the device (Fig. 6).

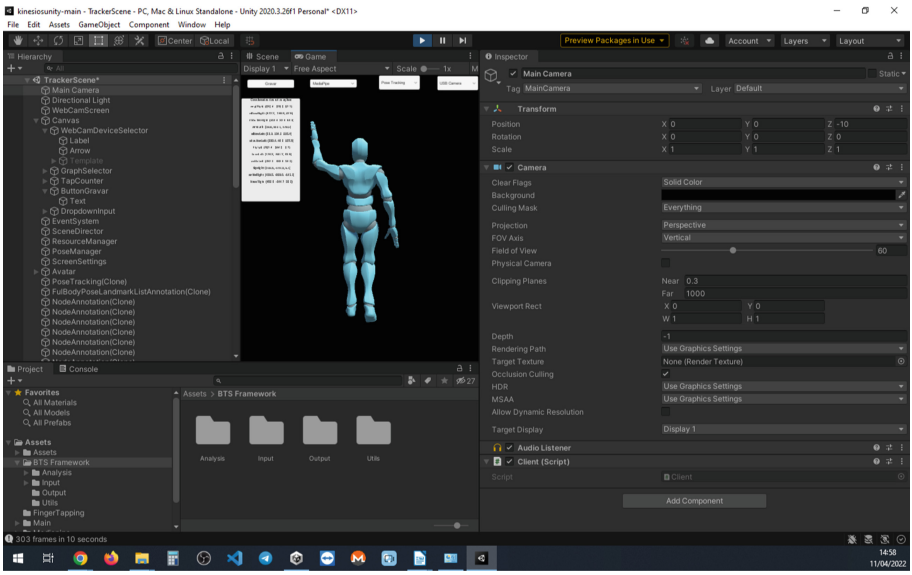

Fig. 6. Example with avatar

5 Conclusion

In this paper, we describe the creation of a framework capable of facilitating the creation of body tracking applications aimed at neuromotor rehabilitation.

We compared two development approaches in our tests: without the framework and with the framework. We could observe that it took much more time to achieve the same result. Besides that, several other factors made the development of a quality application difficult, highlighting: the high coupling of the code without the framework, the lack of construction of a project following the concepts of software engineering, poor documentation, and difficulty of reuse.

The framework is currently only available on the private GitHub of our research group. Later, the framework will be made available on GitHub for everyone so that more people can contribute to the work.

Health professionals are already testing some works of our research group. We hope that with the adoption of the framework, more works can be developed with better code quality, greater reuse, better maintenance, and increased productivity so that more software systems can reach patients, thus having a more significant social impact and at the same time collecting more data that can help in future treatments.

The captured data can help bring important information on how patients behave to specific treatments. The data can help develop better treatments and help evolve the systems developed. In the future, machine learning algorithms will be applied to this data.

References

1. On-device, real-time hand tracking with mediapipe (2019). https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html
2. Amiri, A.M., Shoaib, N., Hiremath, S.V.: A framework to enhance assistive technology based mobility tracking in individuals with spinal cord injury. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 467–471. IEEE (2017)
3. Barnes, T., Encarnação, L.M., Shaw, C.D.: Serious games. IEEE Comput. Graph. Appl. **29**(2), 18–19 (2009)
4. Braga, R.T.V.: Um processo para construção e instanciação de frameworks baseados em uma linguagem de padrões para um domínio específico. Ph.D. thesis, Universidade de São Paulo (2002)
5. Brandão, A.F.: Biomechanics sensor node for virtual reality: a wearable device applied to gait recovery for neurofunctional rehabilitation. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12255, pp. 757–770. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58820-5_54
6. Brown, L., García-Vergara, S., Howard, A.M.: Evaluating the effect of robot feedback on motor skill performance in therapy games. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1060–1065. IEEE (2015)
7. Chang, R., Lau, S., Sim, K., Too, M.: Kinect-based framework for motor rehabilitation. In: 2016 International Conference on Robotics, Automation and Sciences (ICORAS), pp. 1–4. IEEE (2016)

8. Cifuentes, C.A., Rodriguez, C., Frizera-Neto, A., Bastos-Filho, T.F., Carelli, R.: Multimodal human-robot interaction for walker-assisted gait. IEEE Syst. J. **10**(3), 933–943 (2014)
9. Da Gama, A., Chaves, T., Figueiredo, L., Teichrieb, V.: Poster: improving motor rehabilitation process through a natural interaction based system using kinect sensor. In: 2012 IEEE Symposium on 3D User Interfaces (3DUI), pp. 145–146. IEEE (2012)
10. Dhawan, A.P., Huang, B.H., Kim, D.S.: Principles and Advanced Methods in Medical Imaging and Image Analysis. World Scientific, Singapore (2008)
11. Drobnjakovic, F., et al.: Fusing data from inertial measurement units and a 3D camera for body tracking. In: 2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), pp. 1–6. IEEE (2018)
12. Duarte, N., Postolache, O., Scharcanski, J.: KSGphysio-kinect serious game for physiotherapy. In: 2014 International Conference and Exposition on Electrical and Power Engineering (EPE), pp. 606–611. IEEE (2014)
13. Ferreira, I.A., de Resende, A.M.P., Costa, H.A.X.: Análise do impacto da aplicação de padrões de projeto na manutenibilidade de um sistema orientado a objetos. Monografia de Graduação do Departamento de Ciência da Computação da Universidade Federal de Lavras, 78p (2012)
14. Forsyth, D.A., Ponce, J.: Computer Vision: A Modern Approach. Pearson, London (2012)
15. Fowler, M.: Padrões de arquitetura de aplicações corporativas. Bookman (2009)
16. Gamma, E.: Design Patterns. Pearson Education India (1995)
17. Gamma, E.: Padrões de projetos: soluções reutilizáveis. Bookman editora (2009)
18. Gauthier, S., Cretu, A.M.: Human movement quantification using Kinect for in-home physical exercise monitoring. In: 2014 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pp. 6–11. IEEE (2014)
19. Gavrila, D.M.: The visual analysis of human movement: a survey. Comput. Vis. Image Underst. **73**(1), 82–98 (1999)
20. Gordon, C., Roopchand-Martin, S., Gregg, A.: Potential of the nintendo wii™ as a rehabilitation tool for children with cerebral palsy in a developing country: a pilot study. Physiotherapy **98**(3), 238–242 (2012)
21. Van de Graaff, K.M.: Anatomia humana. In: Anatomia Humana, p. 840 (2003)
22. Gwak, M., et al.: EXTRA: exercise tracking and analysis platform for remote-monitoring of knee rehabilitation. In: 2019 IEEE 16th International Conference on Wearable and Implantable Body Sensor Networks (BSN), pp. 1–4. IEEE (2019)
23. Kitsunezaki, N., Adachi, E., Masuda, T., Mizusawa, J.I.: Kinect applications for the physical rehabilitation. In: 2013 IEEE International Symposium on Medical Measurements and Applications (MeMeA), pp. 294–299. IEEE (2013)
24. Kurillo, G., Han, J.J., Nicorici, A., Bajcsy, R.: Tele-MFAsT: kinect-based telemedicine tool for remote motion and function assessment. In: MMVR, pp. 215–221 (2014)
25. Macknojia, R., Chávez-Aragón, A., Payeur, P., Laganière, R.: Calibration of a network of kinect sensors for robotic inspection over a large workspace. In: 2013 IEEE Workshop on Robot Vision (WORV), pp. 184–190. IEEE (2013)
26. Maggiorini, D., Ripamonti, L., Scambia, A.: Videogame technology to support seniors. In: Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, pp. 270–277 (2012)

27. Maggiorini, D., Ripamonti, L.A., Zanon, E.: Supporting seniors rehabilitation through videogame technology: a distributed approach. In: 2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS), pp. 16–22. IEEE (2012)
28. Myagmarbayar, N., Yuki, Y., Imamoglu, N., Gonzalez, J., Otake, M., Yu, W.: Human body contour data based activity recognition. In: 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 5634–5637. IEEE (2013)
29. de Oliveira, C.E.N., Salina, M.E., Annunciato, N.F.: Fatores ambientais que influenciam a plasticidade do snc. Acta Fisiátrica **8**(1), 6–13 (2001)
30. Patterns, D.: Erich gamma, richard helm, ralph johnson, john vlissides. 2003 (1994)
31. Postolache, O.: Physical rehabilitation assessment based on smart training equipment and mobile apps. In: 2015 E-Health and Bioengineering Conference (EHB), pp. 1–6. IEEE (2015)
32. Rodriguez-Martin, D., Sama, A., Perez-Lopez, C., Catala, A., Cabestany, J., Rodriguez-Molinero, A.: SVM-based posture identification with a single waist-located triaxial accelerometer. Expert Syst. Appl. **40**(18), 7203–7211 (2013)
33. Sinha, S., Bhowmick, B., Sinha, A., Das, A.: Accurate estimation of joint motion trajectories for rehabilitation using Kinect. In: 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 3864–3867. IEEE (2017)
34. Stütz, T., et al.: An interactive 3D health app with multimodal information representation for frozen shoulder. In: Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 1–11 (2017)
35. Vaghetti, C.A., Duarte, M.A., Ribeiro, P.O., Botelho, S.S.: Using exergames as social networks: testing the flow theory in the teaching of physical education. In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE), vol. 23 (2012)
36. Whitall, J., Waller, S.M., Silver, K.H., Macko, R.F.: Repetitive bilateral arm training with rhythmic auditory cueing improves motor function in chronic hemiparetic stroke. Stroke **31**(10), 2390–2395 (2000)