Framework for registration and recognition of free-hand gestures in digital games

Tiago Ramos Ribeiro[1]* Daniela de Sousa Costa [1] Polyana Bezerra da Costa [1]
Petterson de Sousa Diniz [1] Paulo Roberto Jansen dos Reis [1] Aristófanes Corrêa Silva [2]
Geraldo Braz Junior [1] Anselmo Cardoso Paiva[1]

[1]Federal University of Maranhão, Department of Informatics, Brazil
[2]Federal University of Maranhão, Department of Electrical Engineering, Brazil

ABSTRACT

This paper presents a framework that aims to simplify the development of digital games based on gestural interaction with the Leap Motion device (LM). The framework's architecture allows the registration of a set of free-hand gestures and enables their recognition in whichever game its used. Based on the natural user interface (NUI) paradigm, the framework to be presented promotes a simple and intuitive way of interaction between the final user and the application, allowing developers to use it without extensive knowledge of the device's hardware and its SDK, due to the application's level of abstractness. Using this approach, developers only have to relate gestures to events. In order to guarantee loose coupling between the framework and the client's application, it was applied the design pattern Publisher/Subscriber, where one does not need to know the existence of the other. Furthermore, as a case of study, it will be presented the integration of the proposed framework to a wheelchair game built on Unity Engine.

Keywords: Natural User Interface, Leap Motion, Unity, gesture interaction

1 INTRODUCTION

The wide availability of devices capable of tracking the whole body for interaction (such as Microsoft Kinect, Creative Interactive Gesture Camera, Leap Motion) has enabled a pervasive introduction of gestural interaction in our everyday life. This incorporates more natural ways for interaction, named Natural User Interface (NUI).

The natural user interface paradigm aims to reduce the barrier between user and application, resembling the way people interact with things in their daily activities. According to [18], the word "natural" implies that the final user will be capable of using the interface or interacting with the software with minimum to no training. Gestures are the most primary and expressive form of human communication, thus it must be also useful for human computer interaction [20].

Hand gestures provides a natural and convenient way of human–computer interaction (HCI) and they are one of the most used forms of interaction in the NUIs paradigm. NUIs allow interaction with computer applications in a manner which mimics the physical world. When used properly they are easy to understand by novice users. Free-hand interaction has been used in various applications such as 3D modeling in immersive environments [8], augmented reality applications [5], writing and sketching [19], etc. Recently it has become popular in the video game industry.

The richness and intuitiveness of free-hand interaction often comes at the cost of posing a great challenge for developers to construct these gesture based dialogs. The combination of the relative positions of the hand and the fingers, associated with the hand's movement result in a complex model, increasing the difficulty for developers to program, test and iterate with these gestures. To deal with this challenge, it is necessary to provide tools that allow developers to easily incorporate free-hand interaction to their applications. There are central problems to develop these interactions: gesture specification (representation) and recognition (manipulation) of the specified gestures. In general the tools provided by sensors companies are focused on textual programming. The representation problem is in general addressed by three paradigms: use of two-dimensional graphs of the data; use of a a visual markup language; and, use of a a time-line of frames. To implement the manipulation process the representation model may use two approaches: demonstration and declaration. When programming by demonstration the developer describes gestures by example. In general, many examples of the same gesture must be provided in order to account for the differences in gesturing between users and over time. The declarative approach involves the use of a high-level specification language to describe the gesture and recognize it.

This work proposes an alternative approach for free-hand gesture recognition that is based on the use of a simple representation of the hand pose and movements. Thus we propose a framework to help developers to incorporate free-hand interactions in games applications, providing a simple way of interaction with less effort from the user. The framework aims to support the developers who want to use gestural interactions with Leap Motion device, allowing them to integrate the framework to their application easily, without knowing the device hardware and its SDK. The framework is based on single registration of a gesture by an example made using the sensor, and a posterior recognition capability. There is the possibility to work with three different types of gestures and the user is free to choose what kind of them is more suitable to its needs.

The remainder of this paper is organized in another five sections. Section 2 presents related work, projects with purposes similar to ours using Leap Motion and other devices plus a brief comparison between them and the proposed framework. Section 3 presents the paradigms and patterns that composed the basis of this work. In Section 4, the framework itself is presented and each module of it as well. The categories of hand gestures, the application of the publish/subscribe pattern, the recording and recognition module and the device used to track the hands are discussed. As a case of study, the fifth section shows the integration between our framework and a game developed in Unity Engine. The sixth section shows the results of this work, obtained through tests focused on the accuracy of gesture recognition. The final section presents the conclusion and remarks of this work.

2 RELATED WORK

This section presents works published in the literature that are related to the use and recognition of free-hand gestures.

*e-mail: tiago.rr@outlook.com

Gesture Spotting Framework [9] is a gesture recognition systems that do not use Leap Motion. This framework is based on video processing and models. Each gesture is a sequence of smaller units (sub-gestures). The video is defined as a set of zero or more gestures, and aims to find the frame that initiate an finalize each gesture. The classification algorithm uses a Hidden Markov Model (HMM) variation, obtained by its combination with the model of Hidden Conditional Random Fields (HCRF). This approach provides a simple structure to deal with explicit modeling of sub-gestures. Additionally, requires training algorithms computationally less expensive and results in a model with a better accuracy than the obtained by the use of each technique separately.

In [4] is presented a library of functions developed to ease the manipulation of games interfaces based on gestures and voice. Using the hardware Kinect the gestures/poses are registered and stored, generating a catalogue that is queried when a new gesture is detected. Other framework proposal, using Kinect data and computer vision techniques for real time static gestures recognition is presented in [17]. The main application of this proposal, named XKin , is the recognition of American Sign Language alphabet, that include letters that involves hands poses and movements.

A system using Leap Motion is presented in [11]. The device captures depth information and use them to create a model for recognition based on convolutional neural network. To create a set of 12 pre-defined gestures, it was used 100 volunteers. Each volunteer must register a version for each of the 12 gestures.

Other approach was proposed in [10] to combine Leap Motion with Kinect. The proposal uses some information acquired by LM, like angle, distance and finger elevation. From Kinect it uses other information like distance from fingers to hands palm and hands curvature. All these characteristics are subdivided in 5 feature vectors (3 from LM and 2 from Kinect), that are used by an SVM classifier. The test was done using 10 gestures acquired from the two devices, obtaining an accuracy of 91.8%.

Other proposal, that is not restricted to one recognition technique is the Leap Trainer framework [16]. It proposes three strategies for gestures and poses recognition. : Geometric Template Matching, Cross-Correlation and Artificial Neural Networks. Each strategy may be used through the Leap Trainer API. It allows the record and recognition of movements. In this way its expected that developers may explore and also increment the algorithms for these task.

Other proposal that uses different algorithms for gesture recognition is presented in [15]. It uses a movement classification that is based on the gesture type we want to recognize (static or dynamic). For each gesture type there is a different set of features. Static Gestures are recognized using five features and Support Vector Machine (SVM), on the other side dynamic gestures use Hidden Markov Models (HMM) with six classes.

To the best of our knowledge, we see that there is no framework that provides three types of gesture and allows these gestures to be defined by a single session record.

3 Background

This section presents the methods, patterns and the device used as a basis to develop this work: the natural interface paradigm, the publisher/subscriber pattern and the Leap Motion device. A brief explanation of these methods and the Leap Motion device is shown in the subsections below.

3.1 Leap Motion

Leap Motion is a USB device designed to be used in desktops to get hands information. Leap Motion uses three LEDs and two near infrared cameras to acquire frames and a software called Leap Motion Service to process data to get hands information.

The Leap Motion system employs a right-handed Cartesian coordinate system. The origin is centered at the top of the Leap Motion Controller. The x and z axes lie in the horizontal plane, with the x-axis running parallel to the long edge of the device. The y-axis is vertical, with positive values increasing upwards (in contrast to the downward orientation of most computer graphics coordinate systems). The z-axis has positive values increasing toward the user.[1] (Figure 1)

Measure	Unit
Distance	millimeters
Time	microseconds
Speed	millimeters per second
Angle	radian

Table 1: Units of measurement used by Leap Motion

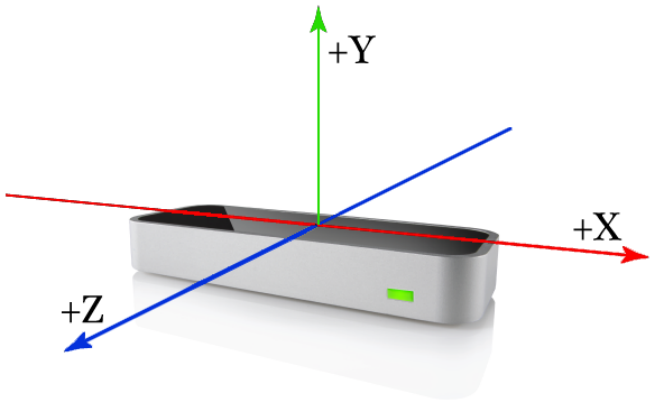

Figure 1: Leap Motion Coordinates System [1]

Table 1 shows the units of the physical quantities measured by Leap Motion.

Leap Motion is able to track hands, fingers and arms in a field of view of one half space and one meter of distance. By providing complete information about the hand, Leap Motion's tracking is very accurate.

We chose to use this sensor in our framework because of its portability and accuracy. Leap Motion is small, light weighted, charged by an USB cable and it can be easily positioned for hand tracking. These factors make Leap motion the ideal device for free-hand gestures recognition.

3.2 Natural User Interface

According to [7], Human-Computer Interaction (HCI) is used to make computer technology more usable by people. The human-computer interaction is a fundamental part of the user's experience in a digital application, especially in games. Nevertheless, the traditional way of interaction through mouses and keyboards aren't natural and in certain applications, such as immersive games, does not contribute to a good experience. In order to guarantee a intuitive and natural interaction, the mouse and the keyboard are being replaced by touch and motion based interfaces, known as Natural User Interfaces (NUI) [3].

The Natural User Interface paradigm was the base to choose the approach of the proposed framework, since it offers a human-computer interaction that resembles the way we interact in our daily routines.

Some of the most used interactions in NUI are listed and explained below, including the one used in our framework.

- Gaze: This HCI is defined as the direction to which the eyes are pointing in space. It is a strong indicator of attention

[6]. Eye tracking systems can be grouped into wearable or non-wearable, and infrared-based or appearance-based. In infrared-based systems, a light shining on eye creates a red-eye effect the difference in reflection between the cornea and the pupil is used to determine the direction of sight. Infrared-based system are very accurate, but long exposure to these kind of light could damage the eyes. In appearance based systems, computer vision techniques are used to find the eyes in the image and then determine their orientation. Wearable systems are the most accurate but they are also very invasive and non-wearable system often requires calibration [7];

- Facial expressions recognition: The detection of facial expression consists on the basic movements of facial features. They are called action units (AUs). A facial expression is a high level description of facial motions represented by regions or feature points called action units. The most used approaches on Facial expressions recognition are: the feature-based approach and the region-based approach. The feature-based approach tries to detect and track specific features of the face, such as the corners of the mouth, eyebrows, etc. On the other hand, the region-based approach focus on facial motions in certain regions on the face, such as the eye/eyebrow and the mouth.[7];

- Hand gesture recognition: Gestures are the link between our linguistic abilities and our conceptualizing capacities, according to [12] . So, it's expected that gestures play a especial role in NUI. Especially hand gestures, since the user is already familiar with devices that use hands, such as mouses, keyboards and joysticks.

  Gesture recognition systems in general are divided into these steps [14]:

  1. Image pre-processing: Consists in preparing the video frames for further analysis by extracting clues about the position of the hands. It's also called feature extraction;
  2. Tracking: In this step, the position and other attributes of the hands must be tracked from frame to frame. To extract motion information for recognition of dynamic gestures, it's necessary to distinguish a moving hand from the background and other moving objects, that's why the frame-to-frame tracking is required;
  3. Gesture recognition: Based on the collected data such as the position of the hands and fingers, motion and pose clues, a valid gesture is detected, or not.

Within all these approaches, we chose to use the free-hand gestures interaction. The subsection 4.4 will explain which phases of the gestures recognition showed above are performed by our framework.

4 Framework

4.1 Gesture Types

The proposed framework recognizes 3 types of gesture, a basic gesture and other two gestures derived from the basic one. The basic gesture, named static, consists only in one pose, i.e., a gesture captured in only one frame. To recognize the hand pose the algorithm uses the relative position of the distals phalanges and a fixed position on hand's palm. The figure 2 shows the bones captured by the sensor.

The second gesture type is a dynamic gesture. It is defined by a static gesture and a translation in X or Y axis, i.e., upwards, downwards, to the left or to the right. For this gesture, initially we recognise the hand's pose, a static gesture. When the static gesture is recognized, a time window is opened to wait the translation of the

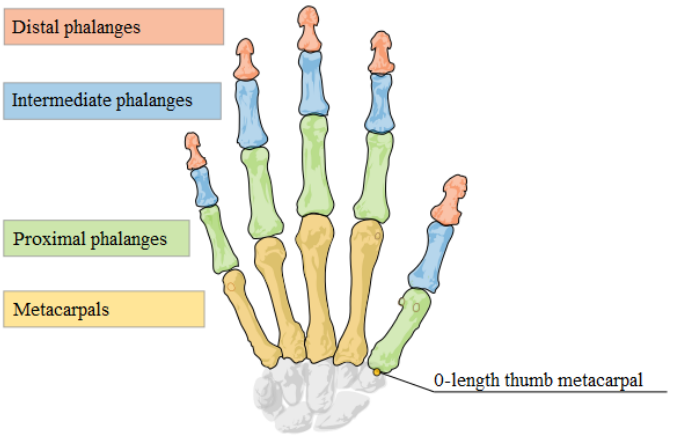

Figure 2: Bones recognized by Leap Motion [1]

Figure 3: A static gesture

hand. If the hand translates in the time window and the translation direction is compatible to the expected, the gesture is recognized. The third gesture, named transition gesture, consists in the detec-

Figure 4: A dynamic gesture

tion of two static gestures inside a time window. When the first gesture is recognized, a time window is opened to wait the second one. If the second one is recognized in the time window, the gesture is recognized.

Figure 5: A transition gesture

4.2 Publish/Subscribe Project Patterns

Microsoft Developer Network (MSDN)[2] defines the Publish/Subscribe as a pattern of project recommended in contexts which there are messages to be sent to other unknown applications, making possible, a superficial relationship. The solution proposed by this pattern involves messages/subjects and subscribed applications that are interested in receiving messages. In this way, a mechanism is created and sends specific messages to all subscribed applications.

This pattern has three variations: List-Based Publish/Subscribe, Broadcast-Based Publish/Subscribe and Content-Based Publish/Subscribe. List-Based Publish/Subscribe notifies all interested applications through a list. Broadcast-Based Publish/Subscribe sends the message to a Local Area Network (LAN) and each node verifies if that message was subscribed. Finally, Content-Based Publish/Subscribe sends a message to interested applications based on context.

One of the main benefits of this pattern is the loose coupling because the final application does not need to know the application that is sending the messages. In the same way, the application sending messages does not need to know the messages receivers. In this framework, we choose to implement the List-Based Publish/Subscribe pattern. The system's architecture is presented in the form of a component diagram in Figure 6. The component *DeviceNotificationService* manage the list of all applications interested in the events trigged by *DeviceController*, which provides a higher level of abstraction in the manipulation of the framework, gathering functions of register, calibration, and recognition of gestures. When a event is trigged, *DeviceController* notifies *DeviceNotificationService* that transmit this notification to all subscribers. *ApplicationNotificationService* manage the notifications of the clients. The notifications are limited by the framework's limitations, e.g., requests of gesture capture, calibration and enable/disable of the recognizer.

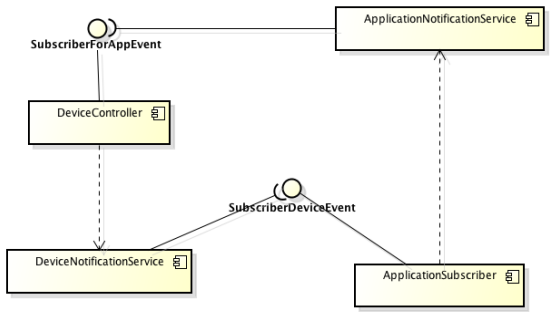

Figure 6: Component diagram.

In addiction to the signature, an observer component needs to implement the interface required by the publish component. All client applications need to implement the interface *SubscriberDeviceEvent*, which is the interface that encapsulates the notifiable methods.

4.3 Capturing Module

To capture gestures, we developed a module of the framework to manage it. In this module, it's possible to see the current pose, select the type of gesture, give a name to the gesture and finally capture the gesture. When a gesture is captured, the user can save this gesture in a XML file. The name of the file is the same of the gesture and is located in the folder *GestureDatabase*, in same directory of the application. This folder can be used in a future application that uses a gesture database. The framework can load this database, if it is in the same folder of the application that will use it.

It is possible to register a gesture by script, using a function of the framework with all the parameters that was aforementioned. Figure 7 shows the capturing module in action.

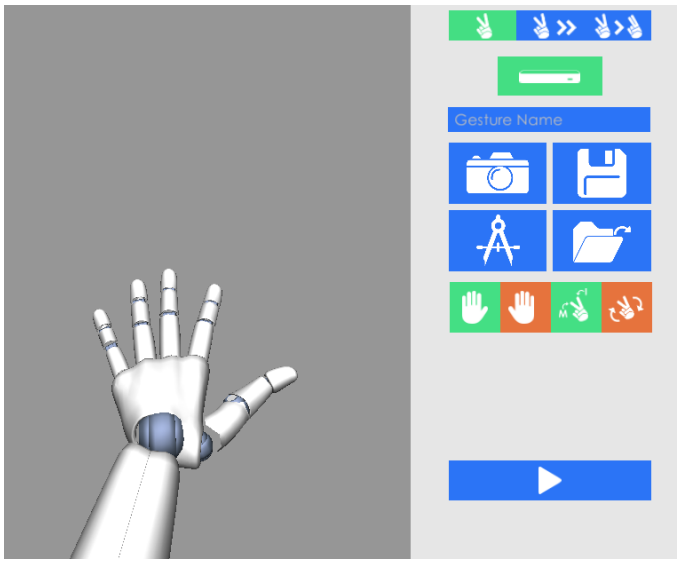

Figure 7: Example of a gesture being captured inside the framework

4.4 Recognition Module

The subsection 3.2 showed that in general, the gesture recognition process pass through a phase of image pre-processing, tracking and the recognition of the gesture itself. In our framework, the image processing phase and the tracking phase is performed by the Leap Motion sensor. The device analyzes the image and extracts hand data. Our framework collect these data and interpret it to discover if the gesture performed is in the gesture database.

First, the gesture's database is loaded and the performed gesture is compared to each gesture of the set. Our framework interprets the data provided by the sensor and extracts the features to categorize the received data according to the types of gestures shown in the subsection 4.1.

To recognize a performed gesture, it is necessary to set which fingers are active and the distances between them. An active finger is a non-folded finger. To determinate whether if a finger is folded or not, the distance of the distal phalange, i.e., the tip of the finger to the palm of the hand is evaluated. After making the match of fingers, the palm normal is verified to define the hand orientation.

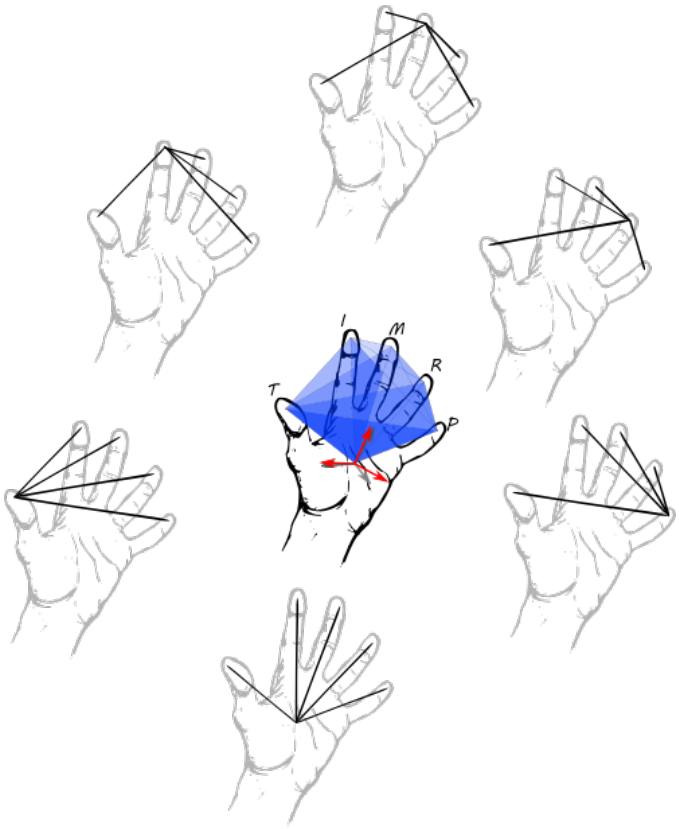

Figure 8: Features analysed to make the match of a pose. The labels T, I, M, R e P stands for Thumbs, Index, Middle, Ring and Pinky finger.

In case of dynamic gestures, a time window is opened to wait for the hand translation. The translation is processed and the direction to which the hand is moving is determined. In order to classify transition gestures, if the first pose matches an already existent gesture, a time window is opened to wait for the second pose. The figure 8 shows some of the features required to make the pose matching.

If all these features match to a saved gesture, the performed pose is recognized and classified. The figure 9 shows a person testing the recognition module. The user performed a gesture already saved in gesture's database.

The figure 10 shows the recognition module interface.

5 Study case of the integration into the wheelchair game

The framework was integrated into a serious game [13] developed in Unity3D. This game aims to educate the player about the problems faced by wheelchair users around the city. In the game, the player controls a wheelchair with the commands go forward, go backwards, turn left, turn right, brake, speed up and slow down. The user is free to register its own set of gestures and relate them to wheelchair commands.

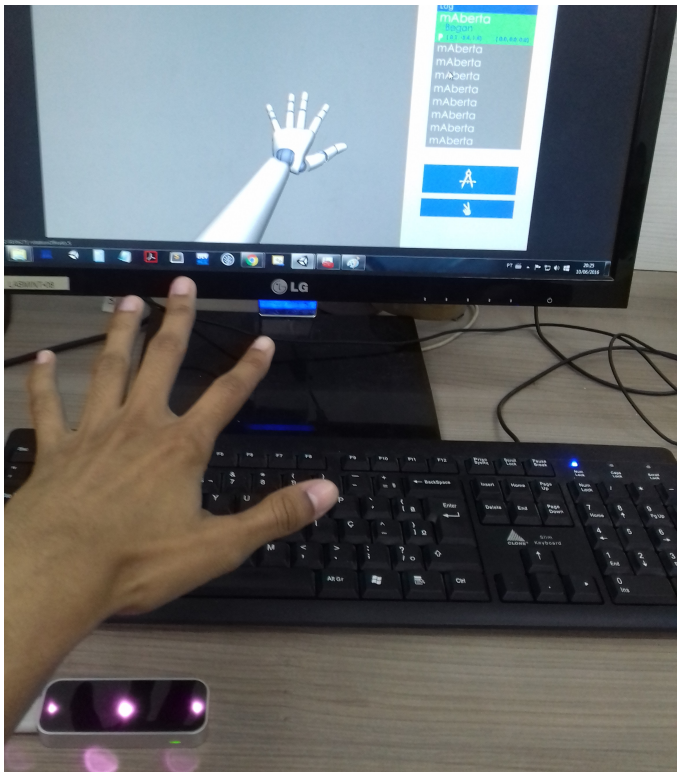

Figure 9: Example of a user performing a gesture in the recognition module.

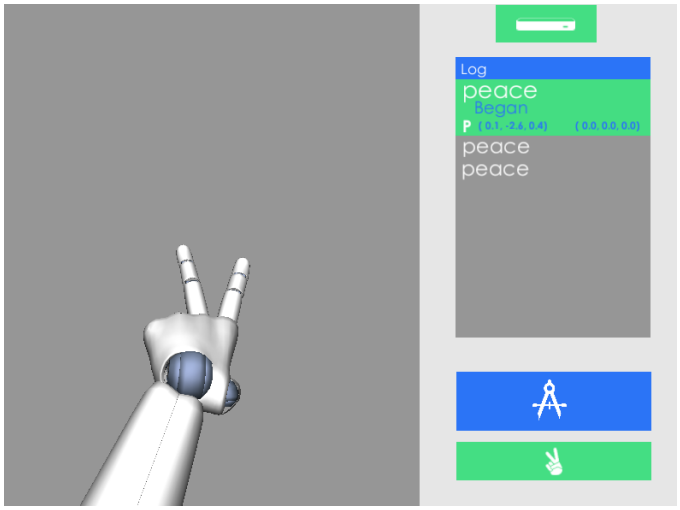

Figure 10: The interface of the recognition module. The text box shows the name and the state of the recognized gestures.

For each command there is an associated gesture, that should be recognized and notified to the game when executed. The integration of the framework and the game was accomplished by adding configuration settings for gesture registration, device calibration and gesture recognition to the game. Nevertheless, the player can also use the keyboard as input.

A survey (Fig. 11) was applied to a group of developers of the game. Using the Likert Scale, this survey aims to evaluate the difficulty level faced by a four-person team to integrate each function provided by the framework. For the first question, the maximum value obtained in the range was 4 for 25% of the development team, which reveals that the majority doesn't qualify as an expert in LM. As for the second question, half of the team gave score 2 for the difficulty in capturing module coupling, while the rest of the team was divided into scores 1 and 3. Regarding calibration and recognition, most of the team (75%) considered the integration of the calibration module as very easy, and 100% gave score 2 to the coupling of the recognition's component . Finally, the evaluation of the effort generated by the framework integration received scores 2 and 3, each representing the opinion of 50% of the team. In general, the difficulty's level of the integration process earned a medium score.

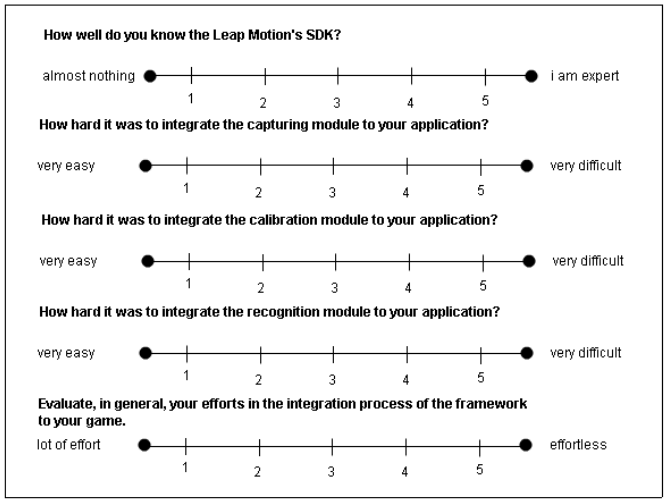

Figure 11: The survey's structure

The figure 12 shows a frame of the game where the user controls the wheelchair through a pre-recorded gesture.

6 Tests and Results

In order to analyze the accuracy of the framework, several tests focused on the recognition module were performed. An instance of the framework and a Leap Motion device were placed on a local school and the volunteers were able to perform a pre-recorded set of gestures. About 10 volunteers were part of the testing process. The set of gestures and their types are listed in the paragraphs below, as well as the accuracy of each gesture. For each type of gesture, the volunteers performed a set of five different gestures, making a total of fifty tests per category.

Concerning the Static type of gesture, the chosen repertoire was: 1- an open hand(all fingers active); 2- a closed hand, but the index finger is active; 3- a closed hand, but the thumbs is up; 4- a closed hand; 5- a closed hand, but the thumbs and index finger are active (an active finger is a non-folded one). The Open hand gesture and the Thumbs up gesture were recognized every time they were performed. They obtained an accuracy of 100%. The Index finger active gesture obtained the second best accuracy, 98%. The closed hand gesture obtained an accuracy of 92%. At last, the lowest accuracy belongs to the Thumbs and index finger active gesture, making a total of 60% of accuracy in the recognition module.

Regarding the Dynamic gestures, the chosen set is listed with the pose and its direction: 1- an open hand with the fingers spread, hand moving leftwards; 2- a closed hand, but the index finger is active and the hand is moving to the right; 3- thumbs, 4- index and pinky finger active, hand moving downwards; 5- a closed hand moving upwards; thumbs, index and middle finger active, hand moving downwards. The first and the last gesture obtained the best accuracy of the set, a total of 100%. The second gesture obtained the second best accuracy, 96%. The third gesture obtained 82% of accuracy. The lowest accuracy of the set belongs to the closed hand moving upwards, making a total of 60%.

Figure 12: Example of a user controlling a wheelchair through hand gestures.

At last, in order to analyse the accuracy of the recognition module with transitional gestures, the following repertoire of gestures was chosen: 1- Pose A: a closed hand with the thumbs and pinky finger up, Pose B: closed hand with the pinky finger up; 2- Pose A: a closed hand with the index finger up, Pose B: closed hand with index and middle finger up; 3- Pose A: closed hand with the thumbs, index and pinky finger up, Pose B: index and pinky finger up; 4- Pose A: closed hand with all the fingers spread, Pose B: Closed Hand; 5- Pose A: Fingers Spread, Pose B: closed hand with the thumbs, index and pinky finger up. The best accuracy of the repertoire belongs to the third gesture, making a total accuracy of 100%. The second gesture has the second best accuracy, with 94% of success in the recognition process. The first gesture has 92% of accuracy. The fourth gesture had a recognition accuracy of 88% and the fifth gesture had 78% of accuracy in the in the recognition module.

The performed tests showed promising results. The average accuracy of gestures recognition was 89%. Gestures with a closed hand showed highest error rate, in the overall process. In general, gestures with all fingers spread showed an excellent accuracy, about 100%. The accuracy rate of each type of gesture is presented at the figure 13.

The ergonomy of the performed gestures is beyond the scope of this report, although it is a theme to be analysed in the future.

In conclusion, the table 2 shows a comparison between the proposed framework and the works presented in previous sections. A

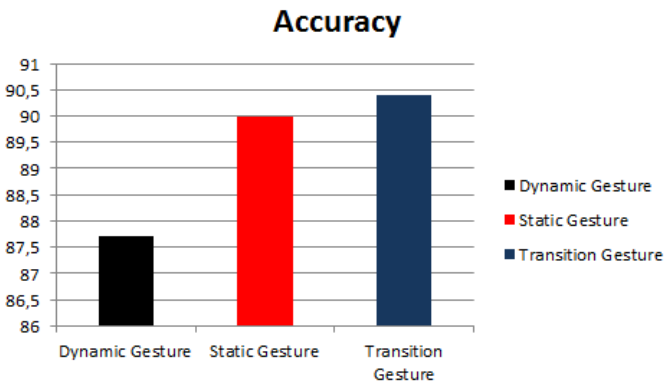

Figure 13: Accuracy of each type of gesture

Comparison of Frameworks								
Refference	[9]	[4]	[17]	[11]	[10]	[16]	[15]	Proposal
LeapMotion	no	no	no	yes	yes	yes	yes	yes
Static Gest.	yes	yes	yes	yes	yes	yes	yes	yes
Dynamic Gest.	no	yes	yes	yes	no	yes	yes	yes
Transition Gest.	yes	yes	no	no	no	no	no	yes
Training	yes	no	yes	yes	yes	yes	yes	no
Reusability	no	yes	no	no	no	yes	no	yes

Table 2: Comparison between Frameworks.

further analisis of these works shows that all of them work with static gestures, 75% of them accept dynamic gestures and only 37,5% of them all accept transitional gestures. We may see that our framework along with [4] have the most complete set of gestures, registering three different types. Nevertheless, the proposed framework has Leap Motion as input device while [4] uses Kinect. Although being an excellent sensor, Kinect does not offer complete information about the hands, such as the tip of the fingers or the fingers itself, requiring even more processing to recognize gestures.

7 Conclusion

This paper presented a framework for registration and recognition of free-hand gestures whose main contribution is the development of a software architecture and a set of algorithms that facilitate the use of gestural interaction in games and digital applications, in order to grant an intuitive interaction. The proposed framework provides gestural interaction in games based on the use of Leap Motion device. An evaluation performed with developers showed that the framework allows easy integration with games, without requiring previous knowledge of the device. Also, it presents a simple approach, based on a single registration of a gesture using the sensor, and a posterior recognition module. Additionally, it serves a rich set of possible gestures as it is also possible to work with three different types of gestures, providing the developers agility and a great flexibility in the gesture interaction design.

The recognition accuracy of the gesture presented good results and it is greater when the gesture is performed with a distance from one hand to another. When compared with other frameworks proposed in the literature we may see that the presented work offers a more complete set of different types of gestures.

As future work we intend to extend the framework to work with other devices like Intel Depth camera and data gloves. Also there is a need to adjust the recognition algorithm for some gestures templates that presented smaller accuracy in the tests.

Acknowledgements

The authors acknowledge the financial support provided by CNPq and FAPEMA.

References

[1] *Leap Motion Overview*. Available at: https://developer.leapmotion.com/documentation /csharp/devguide/Leap_Overview.html. Accessed: 2016-06-08.

[2] *Publish Subscribe*. Available at: https://msdn.microsoft.com/en-us/library/ff649664.aspx. Accessed: 2016-06-08.

[3] A. Câmara. Natural user interfaces. In *Human-Computer Interaction–INTERACT 2011*, pages 1–1. Springer, 2011.

[4] G. S. Cardoso, A. Schmidt, and C. da Computaçao. Biblioteca de funções para utilização do kinect em jogos eletrônicos e aplicações nui. In *XXVI conference on graphics, pat terns and images (august 2013, Arequipa, Peru)*, 2012.

[5] D. Datcu and S. Lukosch. Free-hands interaction in augmented reality. In *Proceedings of the 1st Symposium on Spatial User Interaction*, SUI '13, pages 33–40, New York, NY, USA, 2013. ACM.

[6] A. T. Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470, 2002.

[7] A. Jaimes and N. Sebe. Multimodal human–computer interaction: A survey. *Computer vision and image understanding*, 108(1):116–134, 2007.

[8] H. Kim, G. Albuquerque, S. Havemann, and D. W. Fellner. Tangible 3d: Hand gesture interaction for immersive 3d modeling. In *Proceedings of the 11th Eurographics Conference on Virtual Environments*, EGVE'05, pages 191–199, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[9] M. R. Malgireddy, J. J. Corso, S. Setlur, V. Govindaraju, and D. Mandalapu. A framework for hand gesture recognition and spotting using sub-gesture modeling. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3780–3783. IEEE, 2010.

[10] G. Marin, F. Dominio, and P. Zanuttigh. Hand gesture recognition with leap motion and kinect devices. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 1565–1569. IEEE, 2014.

[11] R. McCartney, J. Yuan, and H. Bischof. Gesture recognition with the leap motion controller. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, page 3. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[12] D. McNeill. *Hand and mind: What gestures reveal about thought*. University of Chicago Press, 1992.

[13] S. A. Melo, G. S. Drummond, D. de Sousa Costa, P. B. Costa, M. A. M. Silva, I. M. Maia, J. V. M. Figueiredo, and A. C. Paiva. Design de um jogo sobre problemas de acessibilidade enfrentados por cadeirantes. 2015.

[14] T. B. Moeslund and L. Norgard. A brief overview of hand gestures used in wearable human computer interfaces. *Computer Vision and Media Technology Lab., Aalborg University, DK, Tech. Rep*, page 8, 2003.

[15] M. Nowicki, O. Pilarczyk, J. Wasikowski, K. Zjawin, and W. Jaskowski. Gesture recognition library for leap motion controller. *Bachelor thesis. Poznan University of Technology, Poland*, 2014.

[16] R. O'Leary. Leap trainer framework. https://github.com/roboleary/LeapTrainer.js/tree/mastersub-classingthe-leaptrainercontroller, 2015.

[17] F. Pedersoli, S. Benini, N. Adami, and R. Leonardi. Xkin: an open source framework for hand pose and gesture recognition using kinect. *The Visual Computer*, 30(10):1107–1122, 2014.

[18] G. Steinberg. Natural user interfaces. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[19] S. Vikram, L. Li, and S. Russell. Writing and sketching in the air, recognizing and controlling on the fly. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 1179–1184, New York, NY, USA, 2013. ACM.

[20] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Commun. ACM*, 54(2):60–71, Feb. 2011.