Hand Gesture Input Interface of *IntelligentBox* Using Leap Motion Controller and Its Application Example

Takumi Takeshita[1], Kosuke Kaneko[2], and Yoshihiro Okada[1,3](✉)

[1] Graduate School of ISEE, Kyushu University Library, Kyushu University, Fukuoka, Japan
okada@inf.kyushu-u.ac.jp

[2] AiRIMaQ, Kyushu University Library, Kyushu University, Fukuoka, Japan

[3] ICER (Innovation Center for Educational Resources), Kyushu University Library, Kyushu University, Fukuoka, Japan

Abstract. In this paper, the authors introduce hand gesture input interface of *IntelligentBox* and show its application example. *IntelligentBox* is a component-based constructive 3D graphics software development system. One of its application fields is VR (Virtual Reality). VR applications should support various types of VR peripherals, e.g., HMD (Head Mounted Display), data-gloves and so on. *IntelligetBox* supports most of them with its dedicated software components. As a hand gesture input interface, *IntelligentBox* supports data gloves. However, Leap Motion Controller is more portable than data-gloves because data-gloves request the user to wear them on his/her hands although Leap Motion controller does not so. Therefore, the authors added new functionality to support Leap Motion Controller for the hand gesture input interface of *InelligentBox*. In this paper, the authors show an animation system using Leap Motion Controller as one of the applications developed using *IntelligentBox*.

1 Introduction

In recent years, VR (Virtual Reality) technologies have made remarkable progress and VR applications have become widespread. To develop VR applications, we need any development systems. Our research group has already proposed *IntelligentBox* [1] that is a component-based constructive 3D graphics software development system. This system can be used as a development system for VR applications [2–6] because it supports various VR peripherals. As a hand gesture input interface, *IntelligentBox* supports data-gloves. However, Leap Motion Controller is more useful because it is a non-touch hand gesture input interface and cheaper than data-gloves. Therefore, we added new functionality that supports Leap Motion Controller in *IntelligentBox*. In this paper, we explain the functionality and how it works as a hand gesture input interface for I*ntelligentBox*. As an application example, we introduce one animation system.

The remainder of this paper is organized as follows: Sect. 2 describes related work. In this section, we also explain the essential mechanism of *IntelligentBox* and the detail of Leap Motion Controller. Section 3 introduces a hand gesture input interface of *IntelligentBox* and its animation system using Leap Motion Controller. Then, we discuss some problems in Sect. 4. Finally, we conclude the paper in Sect. 5.

L. Barolli (Ed.): BWCCA 2021, LNNS 346, pp. 139–147, 2022.
https://doi.org/10.1007/978-3-030-90072-4_14

2 Related Work

Our research purpose of *IntelligentBox* is to propose a software architecture that makes it easier to develop 3D graphics applications including VR applications. Its related works are 3D graphics toolkit systems and programming libraries like Open Inventor [7], Coin3D [8] and 3D Widget [9]. Open Inventor is an OpenGL based object-oriented programming library. Coin3D is also library very similar to Open Inventor. 3D Widget is a Widget-based toolkit system for the 3D GUI development. Some of them provide an authoring tool that enables to design 3D graphics contents. Even using such authoring tools, it is not easy to develop 3D graphics applications because developers have to write text-based programs for that. For the development of 3D graphics games, there are several popular game engines, e.g., Unity 3D [10] and Unreal Engine [11]. These game engines have very powerful functionalities. However, they request developers to make any text-based programs.

Our research system *IntelligentBox* and its web version (WebIB) provide various 3D software components called boxes represented as visible, manually operable, and reusable functional objects. Furthermore, they provide a dynamic data linkage mechanism called slot connection as described in the next section. These features make it easier for even end-users to develop 3D graphics applications including web 3D contents without writing any text-based programs. This is the main difference of *IntelligentBox* and *WebIB* from the others. In this paper, especially we propose the hand gesture input interface for making *IntelligentBox* more useful.

2.1 Essential Mechanism of *IntelligentBox*

This subsection explains the essential mechanism of *IntelligentBox* that is a data-linkage mechanism called slot-connection. Figure 1 illustrates a data linkage example among *boxes*. Each *box* has multiple slots those are internal variables of the Box related to its functionality. Its one slot can be connected to one of the slots of the other *box*. This connection is called a slot connection. The slot connection is carried out by three standard messages, i.e., a set message, a gimme (give me) message and an update message, when there is a parent-child relationship between two *boxes*. These messages have the following formats:

(1) Parent *box* set <slotname> <value>.
(2) Parent *box* gimme <slotname>.
(3) Child *box* update.

A <value> in a format (1) represents any value, and a <slotname> in formats (1) and (2) represents a user-selected slot of the parent *box* that receives these two messages. A set message writes a child *box* slot value into its parent *box* slot. A gimme message reads a parent *box* slot value and sets it into its child *box* slot. Update messages are issued from a parent *box* to all of its child *boxes* to tell them that the parent *box* slot value has changed. By these three messages, the two slots of a child *box* and its parent *box* are connected and their two functionalities are combined.

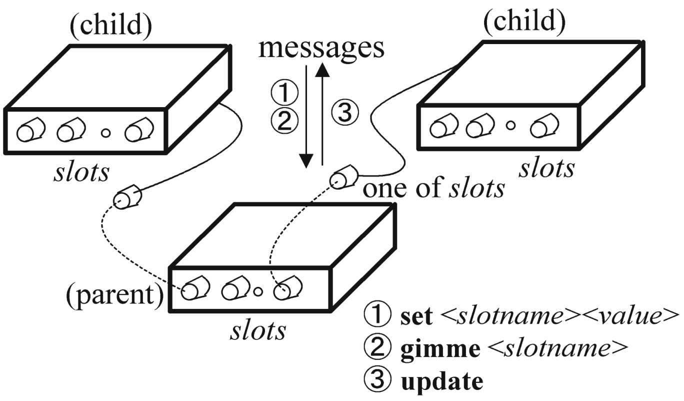

Fig. 1. Standard messages between *boxes.*

2.2 Leap Motion Controller

Figure 2 shows the image of Leap Motion Controller. It is an input interface that can capture hand and finger movements. The software development kit (SDK) is available for C, C#, C++, Java, JavaScript, Python, and Objective-C (currently only Leap C and official C# are supported). They also provide plug-ins that can be used with Unity and Unreal, as well as an application that converts the hand into a cursor on the screen for manipulation [12]. In addition, when the Leap Motion Controller is connected to a PC, it automatically sets up a WebSocket server with port number 6437 on localhost and performs socket communication to output a json data. Users can implement their own applications by communicating with it and getting the json data from it.

Fig. 2. Leap Motion Controller.

The Leap Motion SDK can acquire not only the coordinates of the palm of the hand and the tips of the fingers, but also the coordinates of their bones. In this research, we used this function to obtain the angle of the finger.

3 Hand Gesture Input Interface of *IntelligentBox* Using Leap Motion Controller

As one of the applications of *IntelligentBox* using Leap Motion Controller, we developed a real-time animation system shown in Fig. 3. This application is very similar to our previous animation system [13, 14]. The differences between them are interface devices. The previous system's interface is a data-glove.

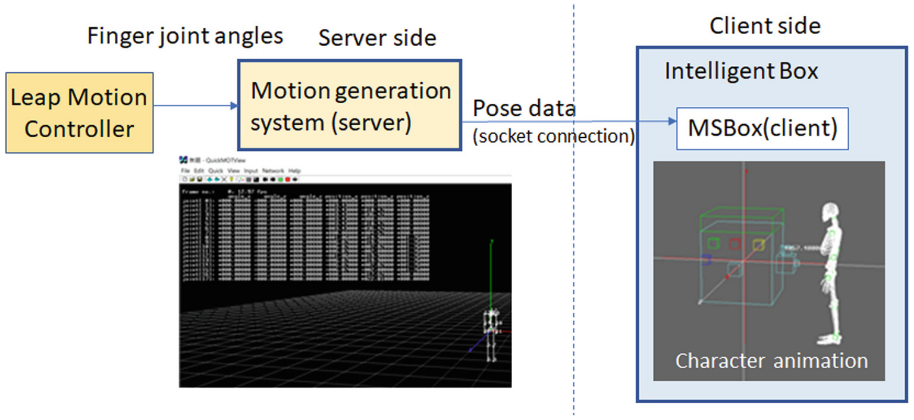

Fig. 3. Component-structure of an animation system of *IntelligentBox*.

3.1 Implementation of the Input Interface

In order to implement the interface, we had to do some verification and programming. The language used for the programming was C++. The following is a description of the process.

3.1.1 Measurement of Finger Joint Angles

First of all, it was necessary to obtain the angle of the joint using Leap Motion Controller, however the function to obtain the angle of the joint is not implemented in the SDK library of Leap Motion. We implemented it using the function to get the coordinates of the tip of the bone. The implementation is as follows:

The 3D coordinates A, B, and C of the three points are received in the form of vectors, and $\overrightarrow{BA}$, $\overrightarrow{BC}$ are obtained from them, and the angle (=θ) is obtained using the inner product formula, as shown in the following equation.

$$\overrightarrow{BA} \cdot \overrightarrow{BC} = \left|\overrightarrow{BA}\right| \times \left|\overrightarrow{BC}\right| \times \cos\theta$$

$$\cos\theta = \frac{\overrightarrow{BA}\cdot\overrightarrow{BC}}{\left|\overrightarrow{BA}\right| \times \left|\overrightarrow{BC}\right|}$$

$$\theta = \cos^{-1}(\cos\theta)$$

The angle obtained by this formula such as π/2 to π cannot be used as it is. In addition, since the range of angle varies from joint to joint, we measured the maximum and minimum angle of each joint (Table 1).

Table 1. Maximum and minimum values of each joint.

平均	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
max	3.1416	3.14136	3.1416	3.14084	3.1416	3.1416	3.1416	3.14016	3.1416	3.13782
min	2.41184	2.1548	1.40574	1.65822	1.47088	1.84548	1.56192	1.75786	1.50916	1.70336

3.1.2 Conversion of Finger Angles to Skeletal Joint Angles

As shown in Fig. 4, the finger joints of the hand and their corresponding skeletal joints were determined. The angle of movement of each was also determined. In this paper, finger joints [8, 9] (little finger) were not assigned to the skeleton. In order to convert the obtained finger joint angles to skeletal joint angles, the following calculations are made.

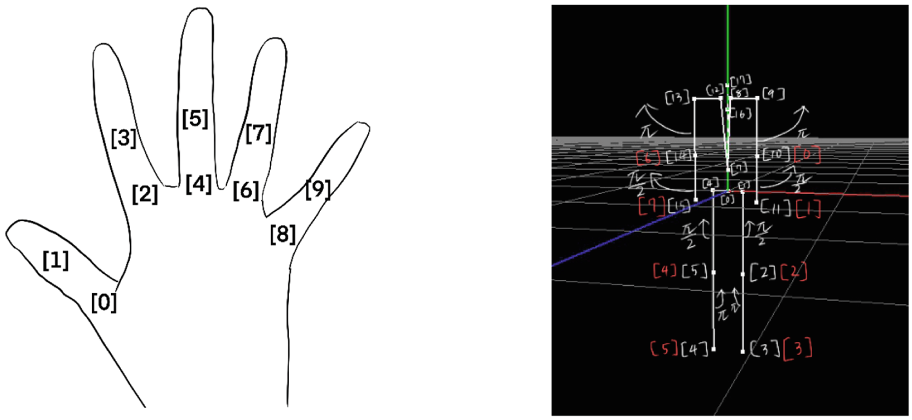

Fig. 4. The left figure shows the numbers of finger joints. The right figure shows the assignment of the finger joints to the skeleton, here the finger joint numbers are red, and the skeletal joint numbers are white.

First, if the angle that had been acquired is larger than the previously set maximum angle or smaller than the previously set minimum angle, correct it to the maximum angle and minimum angle, respectively. The angle of the finger joint (a) is corrected to the angle of the skeletal joint b(0-range)) using the following formula.

$$b = \frac{a - min}{max - min} \times range$$

Since we want the fingers to be open with all the joints extended, that is, the angle of each joint of the skeleton is zero, we can further transform it using the following equation.

$$c = -b + range$$

Since each joint has a parent-child relationship, we get the original (unbent) vector to the child of the joint to be bent. Specifically, we use the rotation formula for vectors in the plane to get the vectors.

$$\overrightarrow{v_2} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \overrightarrow{v_1}$$

The coordinates of the child are determined by adding the vector obtained from this equation to the coordinates of the parent.

3.1.3 Ground Contact

In order to prevent the toes from going down into the underground when the waist of the skeleton model goes down, the system corrects the positions of the knees. For this contact constraint, we use 2-link Inverse Kinematics shown in Fig. 5. In this figure, P_0, P_1 and P_2 mean the base of thigh, the knee and the toe. See the paper [14] for its detail.

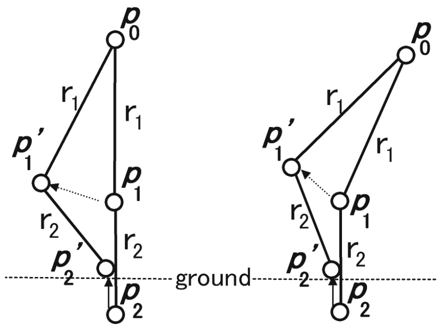

Fig. 5. Ground contact constraint using 2-link Inverse Kinematics.

3.2 Demonstration

Figure 6 shows the actual movement of the skeleton using the Leap Motion Controller on the motion generation system. Figure 7 shows how the skeleton moves on *IntelligentBox*. From these figures, it can be seen that when the fingers are bent, the corresponding joints of the skeleton are also bent, and the Leap Motion Controller is able to control the skeleton model on *IntelligentBox*.

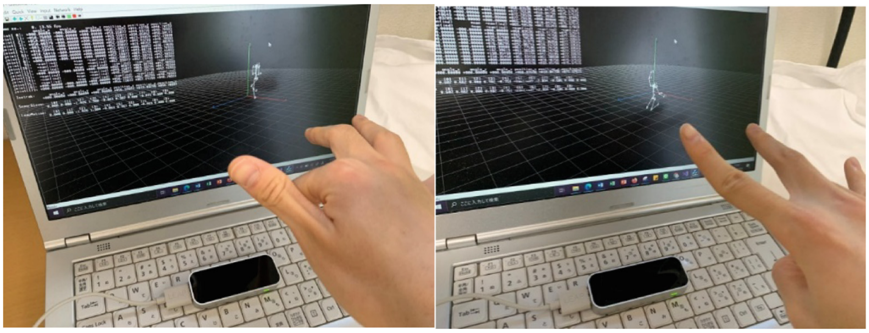

Fig. 6. Poses of the skeleton on the motion generation system, the left knee is going up (left), and the lower legs contact with the ground and the right arm is going up (right).

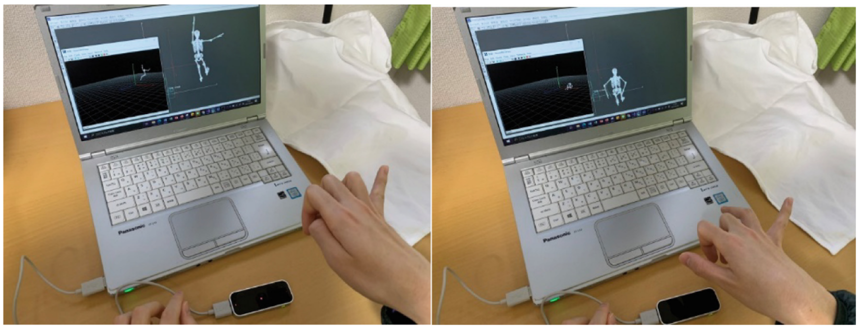

Fig. 7. The poses of the skeleton on *IntelligentBox*, a jump pose (left) and a sitting pose (right).

4 Discussion

4.1 Misrecognition of Leap Motion Controller

In our experiments, joints corresponding to different fingers sometimes moved when the fingers were bent, such as the middle finger being assigned to the ring finger. This is due to the fact that the Leap Motion Controller assigns IDs to fingers in the order in which they are recognized.

To solve this problem, we are now considering a new method of assigning IDs to fingers. We need to modify the program so that it first gets the finger type and then assigns it to the appropriate finger, instead of assigning the joint positions in the order of finger recognition, starting with the thumb.

4.2 Recognition Accuracy of Leap Motion Controller

There were times when I thought I was bending my finger joints firmly in reality, but the Leap Motion Controller did not recognize it, and the fingers did not bend on the screen, or conversely, the skeletal joints on the screen were bent even though I was not bending my fingers. This could simply be due to the Leap Motion Controller not correctly recognizing the position of the finger bones, or the program's maximum and minimum angle settings are too low and the finger angles are not well converted to the skeletal joint angles. To solve this problem, the user can clean the infrared surface of the Leap Motion Controller, adjust the position of the hand so that the Leap Motion Controller can read it easily, or adjust the maximum and minimum angles of the finger joints to make the movement as expected. In some cases, the tracking was lost in the middle of the process, or the skeleton moved suddenly by recognizing different parts as finger bones, or it did not stop steadily. To solve this problem, I felt it was necessary to set an upper limit on the speed so that the joints of the skeleton would move slowly instead of instantaneously, to stabilize the movement by not moving the joints if there is no change in angle beyond a certain level, and to determine the next coordinates based on the average of several frames, and to stabilize the movement by not moving the joint if the angle did not change beyond a certain point.

5 Conclusion

In this research, we developed a system to apply the skeletal movements of the Leap Motion Controller input to *IntelligentBox* via the motion generation system in order to enable the Leap Motion Controller to be used with *IntelligentBox*.

Although there are still many issues to be solved, such as the accuracy of the movements and the range of motion described in Sec. 4, we believe that the results in this paper are promising for the future development of applications using Leap Motion Controller on *IntelligentBox*.

Acknowledgements. This research was partially supported by the project grant from the Kyushu University Education Innovation Initiative.

References

1. Okada, Y., Tanaka, Y.: IntelligentBox: a constructive visual software development system for interactive 3D graphic applications. In: Proceedings of Computer Animation 1995, pp. 114–125 (1995)
2. Okada, Y.: 3D visual component-based approach for immersive collaborative virtual environments. In: Proceedings of the 2003 ACM SIGMM Workshop on Experiential Telepresence, ETP 2003, pp. 84–90 (2003)
3. Okada, Y.: IntelligentBox as component based development system for body action 3D games. In: ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005), pp. 454–457 (2005)
4. Okada, Y., Ogata, T., Matsuguma, H.: Component-based approach for prototyping of Tai Chi-based physical therapy game and its performance evaluations. ACM Comput. Entertainment **14**(1), 4:1–4:20 (2016)
5. Okada, Y., Kaneko, K., Fujibuchi, T.: IntelligentBox based training system for operation of radiation therapy devices. In: Barolli, L., Poniszewska-Maranda, A., Enokido, T. (eds.) CISIS 2020. AISC, vol. 1194, pp. 188–198. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-50454-0_18

6. Yu, B., Shi, W., Okada, Y.: Action input interface of IntelligentBox using 360-degree VR camera and OpenPose for multi-persons' collaborative VR Applications. In: Barolli, L., Yim, K., Enokido, T. (eds.) CISIS 2021. LNNS, vol. 278, pp. 747–757. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79725-6_75
7. Open Inventor. http://oss.sgi.com/projects/inventor/
8. Coin3D. http://www.coin3d.org/
9. 3D Widget. http://www.viewpoint.com/widgets/
10. Unity3D. https://unity.com/
11. Unreal Engine. https://www.unrealengine.com/
12. Ultraleap. https://www.ultraleap.com/product/leap-motion-controller/
13. Okada, Y.: Real-time motion generation of articulated figures using puppet/marionette metaphor for interactive animation systems. In: Proceedings of the 3rd IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP03), pp. 13–18. ACTA Press (2003)
14. Okada, Y.: Real-time character animation using puppet metaphor. In: Nakatsu, R., Hoshino, J. (eds.) Entertainment Computing. ITIFIP, vol. 112, pp. 101–108. Springer, Boston (2003). https://doi.org/10.1007/978-0-387-35660-0_12