Unreal Interactive Puppet Game Development Using Leap Motion

An-Pin Huang¹, Fay Huang¹* and Jing-Siang Jhu¹

¹CSIE, National Ilan University, Taiwan

*fay@niu.edu.tw

Abstract. This paper proposed a novel puppet play method utilizing recent technology. An interactive puppet game has been developed based on the theme of a famous Chinese classical novel. This project was implemented using Unreal Engine, which is a leading software of integrated tools for developers to design and build games. On the other hand, Leap Motion Controller is a sensor device for recognizing hand movements and gestures. It is commonly used in systems which require close-range finger-based user interaction. In order to manipulate puppets’ movements, the developed program employs the Leap Motion SDK, which provides a friendly way to add motion-controlled 3D hands to an Unreal game. The novelty of our project is to replace 3D model of rigged hands by two 3D humanoid rigged characters. The challenges of this task are two folds. First, the skeleton structure of a human hand and a humanoid character (i.e., puppets) are totally different. Making the puppets to follow the hand poses of the user and yet ensuring reasonable puppets’ movements has not been discussed in the literatures nor in the developer forums. Second, there are only a limited number of built-in recognizable hand gestures. More recognizable hand gestures need to be created for the interactive game. This paper reports the proposed solutions to these challenges.

1. Introduction
Puppetry has an ancient history in various cultures with different forms and manipulation methods. In traditional point of view, puppetry is an art performance to deliver stories and/or communicate the ideas among human societies. They are performed in the theaters, special stages, TV shows, or even in the movies. Puppets are controlled by the puppeteers using their hands, arms, or control devices such as rods or strings.

In this project, we proposed a new way to control and manipulate puppets using recent technology. Puppetry nowadays can be performed in virtual world such as using VR gears, or it can be shown by a 3D displaying device or holographic projection. Moreover, puppetry is no longer just a performance by artists, but can also be interactive games accessible to all range of users.

Our interactive game has equipped one Leap Motion (LM) Controller as input device. It is a small USB peripheral device supporting the understanding of hand gestures, analogous to a mouse, but requires no hand contact or touching [1]. It has been widely used for interactive game developments [2, 3]. It is consist of two monochromatic IR cameras and three infrared LEDs. The sensing range is about one meter in distant within a roughly 45-degree pyramid shaped region. The LEDs generate patternless IR light [4] and the cameras generate almost 200 frames per second of reflected data. The 3D position data is then synthesized by comparing the 2D frames generated by the two cameras [5,6].

The most essential spirit of puppetry is the fact that it has to be manipulated by hand gestures, not just...
by pressing some buttons on controllers or keyboards. Thus, Leap Motion device has made this possible due to its capability of sensing fingers movements in real time.

Figure 1. The snapshot of the developed digital puppet game using Leap Motion Controllers as input device.

There are many game engines available in the market. Among them, Unreal Engine 4 (UE4, by Epic Games) and Unity (by Unity Technologies) are arguably two of the most popular game engines for the developments of many well-selling 3D games. But, when it comes to graphics, UE4 is really a next-generation game engine [7, 8]. Another advantage of Unreal Engine (UE) is that it supports a node-based visual scripting method, called Blueprint. C++ is the programming language for UE, but it is possible to create an entire game using only Blueprints. Epic and LM have jointly developed an official plugin for UE4. By taking advantage of this built-in SDK, we are able to extend its capability for developing a novel interactive puppet game.

The goal of this project is to utilize modern technology to carry forward the traditional puppet culture for the young generations. The LM controller is used as a sole input device to detect hand gestures for puppet manipulations. The puppets have been digitalized and represented by 3D models. Figure 1 shows the setup configuration and a snapshot of the game. In order to attract wide range of users, we have adapted the characters from a famous Chinese classical novel, Journey to the West [9], namely Sun Wukong (also known as the Monkey King) and Nezha (also known as Third Lotus Prince). Figure 2 illustrates the 3D models of these two Chinese well-known mythology characters. Two characters can interact with or fight to each other controlled by two separated hands, either from a single user or from two different users. The digital puppets react in real time following the location and orientation of users’ hands like traditional puppets do.

Figure 2. 3D models of Sun Wukong with his Somersault Cloud and Nezha with his Wind-fire Wheels.
2. Methods
This section explains the design of our digital puppet game and elaborates the methods developed using UE4.

2.1. Design of the Interactive Digital Puppet Game
For all forms of puppet shows, audiences are most likely to be attracted to some humorous, rhythmic dialogue and gorgeous battle scenes. In order to deliver excitement and tension during the game play, we have decided to create a combat game. One of the famous battles of “Wukong havoc the Heaven Kingdom” in Journey to the West has been engaged, namely the “Sun Wukong fighting against Nezha”. The Monkey King’s special weapon is called golden cudgel and Nezha in this game is holding a spear. The game is designed that, if one character hit the opponent with his weapon, then it gains points. Within a time-limit, whoever gets the higher point wins the game.

In order to make the game more interesting, some special fighting tricks or martial arts moves for each character can be released by special hand gestures. This idea cannot be realized by traditional puppetry, and thus can be considered as the exclusive extension of traditional puppet play. Other glamorous features only realizable in digital form are all kinds of special visual effects during fighting. Moreover, we have developed particle systems to emulate the well-known accessories belonging to these two characters, they are Wukong’s Somersault Cloud and Nezha’s Wind-fire Wheels (see Fig. 2).

The game flow is depicted in Fig. 3. As the game starts, a countdown is initiated. While the special martial arts moves haven’t been enabled, each player should try to hit the opponent with his weapon. Fold the thumb or pinky to swing the weapon. The special martial arts moves become enabled in a random time manner. When they do, players should compete with each other by correctly completing a set of specified hand gestures to release the special martial arts moves. This would gain more points.

2.2. Introduction of Game Development Component in UE
In UE, the term “level” in UE has similar meaning to a particular status of a virtual world, which can be made up of a collection of game development entities, such as blueprints, static meshes, lightings, particle effects, and volumes etc. Within a defined level, all included entities work together to bring the desired experience to the players.

An “actor” is a generic term for any object that can be placed into a level. There are many different types of actors in UE, for examples, there are player start, camera, skeletal mesh, static mesh, trigger, and particle emitter actors. Actors are a class that support 3D transformations such as translation, rotation, and scale. They can be created and destroyed through gameplay code (blueprints).

A “pawn” in UE is an actor that can be possessed by a Controller, which is set up to easily accept input, and it can do various player-like things. A pawn is not assumed to be humanoid. On the other hand, a “character” extends pawn for a specific human style pawn with specific built-in behaviors.
LM plugin for UE4 by default provides two convenience rigged hand-shape characters, one is a yellow robot hands and the other is white human hands named Echo. We used solely the LeapRiggedEchoHands actor belonged to the Echo character in our project.

2.3. Game Development Method
The developed interactive puppet game involved only one level. A player start actor is included by default placed in the level to designate where the player should start out when the game begins. Two skeletal mesh actors for the characters were newly created in the game for various animations and interactions. Several trigger actors were designed to enable the collision detection and hand gesture determination. Many particle emitter actors were generated in order to create visual effects during game play. This subsection explains some major developments and methods used in this game.

2.3.1. Starting the Game. The game starts when LM device senses two hands (left and right hands) within its sensing range. Two Boolean variables were newly created in the blueprint of LeapRiggedEchoHands actor provided in the LM plugin, one for each hand. While both Boolean variables are true, meaning both hands are recognizable to the LM sensor, the game begins (i.e., the countdown begins).

2.3.2. Replace Echo Hands with Humanoid Characters. By default, Echo hands would be shown, but it is necessary to replace the original hand models by the defined 3D character models. Due to that the rigging (i.e., bone structure) of human hand and humanoid figure are fundamentally different, this is not a trivial task. Making a humanoid figure, in our case the puppet, to follow the user’s hand poses and yet ensuring reasonable human movements has not been discussed in the literatures nor in the developer forums.

The internal technology of LM sensor interprets the rotation value of each finger joint in real time. We have proposed to transform these values into puppet’s joints in order to manipulate the puppet. However, the rotation values do not represent the absolute rotations/orientations with respect to the world coordinate system. Instead, they represent the relative rotation with respect to the hand coordinate system at the middle of two hands. Moreover, due to that fingers’ postures are very different from human’s standard poses. Before those rotation values can be assigned to puppet’s joints, some recalculations (or calibrations) and additional constraints are required. Figure 4 illustrates the proposed binding between a default hand and a puppet’s structures. Only the labeled joints are bound together; moreover, rotation angles were tuned in blueprints to meet the expected puppet movements.

2.3.3. Puppet Animation by Hand Gestures. Two new skeletal actors were created, one for each hand, for the purpose of animating the characters. First of all, they are defined to follow user’s hand location and orientation in real time. Moreover, the poses of the characters should also follow the hand gestures in real time as mentioned in the above paragraphs. New blueprints for these actors were developed to

![Figure 4](image_url). The proposed binding method between a hand and a puppet’s skeleton structures.
enable more variety of hand gestures for different fighting tricks or martial arts moves. We set up trigger actors for all fingers and palm as illustrated by red ellipses in Fig. 5 (left). Different combinations of those triggers define different gestures written in blueprints as shown in the background. Figure 5 (right) depicts the additional gestures specially designed for the game.

**Figure 5.** (Left) Six ellipsoid triggers are added to palm and fingers in accompanied with the created blueprints (only partial view is shown) to recognize hand gestures. (Right) A set of new hand gestures were designed for the game.

### 3. Results

An interactive digital puppet game has been developed from scratch using UE4 (ref. [10]). The 3D models were created mainly in Blender 2.79 in accompany with ZBrush 4R7 for detailed sculpturing. The texturing (UV maps) were enhanced in Photoshop. Normal maps were outputted from Blender. The materials and rendering condition were set up in UE4. Moreover, the animation sequences were generated in Blender and all special effects were created using the particle emitter actors provided in UE4. Figure 6 shows some of the intermediate working details.

The computer is equipped with Windows 7 (64-bit operating system), Intel Xeon E3-1231 V3 (3.4GHz) processor, 16 GB DDR3-1600 memory, and a ASUS GTX660 2GB graphics card. The performance is above 60 fps.

Latest version of the game has been exhibited to public in order to receive more variety of players’ feedbacks. Figure 7 shows some photos of the exhibition. The interaction method has been successfully approved by players of a wide range of ages. Some users have suggested to eliminate few difficult hand postures, and adding more visual effects during the combat. All the advices have been taken into account in the current version. A demo of the current version of the game is available at YouTube (https://youtu.be/Wt8yHfj678M).

### 4. Conclusion

In this paper, a newly developed interactive digital puppet game using recent technology, Leap Motion Controller and Unreal Engine, has been presented. Based on the limited functionalities provided by Leap Motion SDK, extended capabilities were implemented in order to manipulate two humanoid puppet characters played by two players. Additional hand gesture recognitions have been developed to make the puppet interaction more interesting. To the best of our knowledge, the concept and the method of the proposed puppet game is unique and has not been seen or discussed in the literatures nor in the developers’ forums. For the future work, we plan to combine it with one of the available virtual reality displaying methods to bring the players an engaging and more realistic gaming experience.
Figure 6. The interactive digital puppet game has been developed from scratch. Here are some of the intermediate working details,

Figure 7. Some photos from the exhibition.

Acknowledgments
Project mainly funded by Jinjiang Multimedia Technology company (In Chinese:晶將數位多媒體科技); partially supported by MOST 106-2221-E-197-003.

References
[1] Seixas M, Cardoso J and Dias M T G 2015. The Leap Motion movement for 2D pointing tasks: Characterisation and comparison to other devices. In Proc. PECCS'15.
[2] Sutton J 2013. Air painting with Corel Painter Freestyle and the leap motion controller: a revolutionary new way to paint. In Proc. ACM SIGGRAPH '13, Studio talks article No. 21.
[3] Vasudevan S K, Vivek C and Srivathsan S 2015. An Intelligent Boxing Application through Augmented Reality for two users – Human Computer Interaction Attempt. Indian Journal of Science & Technology 8(34) pp 1–7
[4] Weichert F, Bachmann D, Rudak B and Fisseler D 2013. Analysis of the Accuracy and Robustness of the Leap Motion Controller. Sensors 13(5), pp 6380–6393
[5] Guna J, Jakus G, Pogačnik M, Tomažič S and Sodnik J 2014. An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking. Sensors 14(2), pp 3702–3720
[6] Coelho J C and Verbeek F J 2014. Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment. In Proc. Chi Sparks'14. pp 78–85
[7] Petridis, P, Dunwell, I, De Freitas, S and Panzoli, D 2010. An Engine Selection Methodology for High Fidelity Serious Games. In Proc. IEEE VS-GAMES ’10, pp 27–34
[8] Mishra, P and Shrawankar, U 2016. Comparison between Famous Game Engines and Eminent Games. International Journal of Interactive Multimedia & Artificial Intelligence, 4(1), pp 69–77
[9] Wu, C E 2012. The Journey to the West (Vol. 1). University of Chicago Press.
[10] Kanode, C and Haddad, H 2009. Software Engineering Challenges in Game Development. In Proc. IEEE ITNG ’09, pp 260–265