Chess Game System on Smart Mobile Terminal

Dayan Shangguan*, Xintong Li
Beijing Forestry University, Beijing, P. R. China

* Corresponding author. Tel.: 86-010-62338279; email: shanggdy@bjfu.edu.cn
Manuscript submitted October 20, 2014; accepted April 8, 2015.
doi: 10.17706/jsw.10.5.538-550

Abstract: Under the background that traditional Chinese culture needs to be carried forward, the author of this paper, combined with Unity3D game engine, introduces a self-developing Chinese chess game based on Android mobile platform and discusses the basic structure and core mechanics of this game system. Technologies like data transmission in the LAN, Shader of Unity3D, and touch-screen control are adopted in the game design. Through program optimization and compatible design, this game combines 3D modeling tools like Autodesk 3ds Max, utilizes network module of Unity3D and connects two mobile terminals in the LAN, thus finally realizing on-line VS play. Through the Network View component of Unity3D, LAN connection mode suitable for C/S system of Android can be created in Unity3D to achieve data and image synchronization. Test results turn out to prove that with this technology, this game system displays high performance in terms of operation and visual effects.

Key words: Chinese chess, VS system, mobile terminal, Unity3D.

1. Introduction

The popularity of computers not only brings changes to study and work of human beings, but also to our recreation. Recently, computer games, as a recreation form based on computers, has become one of the main recreational activities which can even compare with televisions and films. 3D game, as a nick of games on mobile terminal in Chinese market, has been one of the research priorities in the field of digital game during the recent years. Meanwhile, with the rapid development of game engine technology, a various kinds of commercial engine products emerge into the market, such as Unreal Engine developed by American company Epic, Cry Engine by European company Crytek and Angelica by Chinese company Perfect World [1]-[10]. As an emerging game engine with high performance, Unity3D has been very popular among developers around the world. Nowadays, Unity3D, characterized by its easy-to-use interface, good script compatibility (support C# and JavaScript and realize the invocation of Java through interaction with Activity of Eclipse) and strong rendering power, has become a hot game engine to develop 3D game on mobile phones and turned out to be a guarantee for 3D game on mobile phones to realize high-quality screen effect, smooth operation and resource conservation.

Chinese chess, as one of the important parts of traditional Chinese culture, is regarded as a typical example of VS game of two players [11]-[15]. Although with a long history of development, its popularity has gradually faded with the high pace of modern society. Additionally, most of the existing chess games in the market are not creative or attractive as a result of two-dimensional technology and simple interface [16]. The game system in this paper, as a game of general scientific interest adapted from Chinese chess, combines traditional Chinese culture with 3D technology, thus adding more artistic beauty, playability and

operability. The author hopes to get more people, especially teenagers to be passionate for traditional Chinese culture.

There are a number of research papers about developing technology on mobile phone games. Most of them focus on the researches like game system [17] , [18], structure or optimization of operating speed and less touch upon Chinese chess games. As for the researches on Chinese chess, they mostly center on optimization of storage algorithm, but rarely discuss 3D mobile phone VS games in LAN [19]. This paper, based on a self-developing chess game system on Android mobile terminal, elaborates the implementation scheme of a creative 3D chess game to realize on-line VS play function on Unity3D engine from aspects like system module structure, core mechanics, natural environment and animation simulation.

2. Structure Diagram of Chess Game System

The development environment of this game system is based on Unity3D. 3D algorithm will be adopted to do the preliminary modeling, Photoshop to do the mapping, 3DMax to do the rendering and finally the modeling will be imported to Unity3D for editing and coding [20], [21].

As shown in fig.1, this game system consists of six modules, named as UI module, animation control module, chess core module, camera control module, touch-screen control module and network connection module.

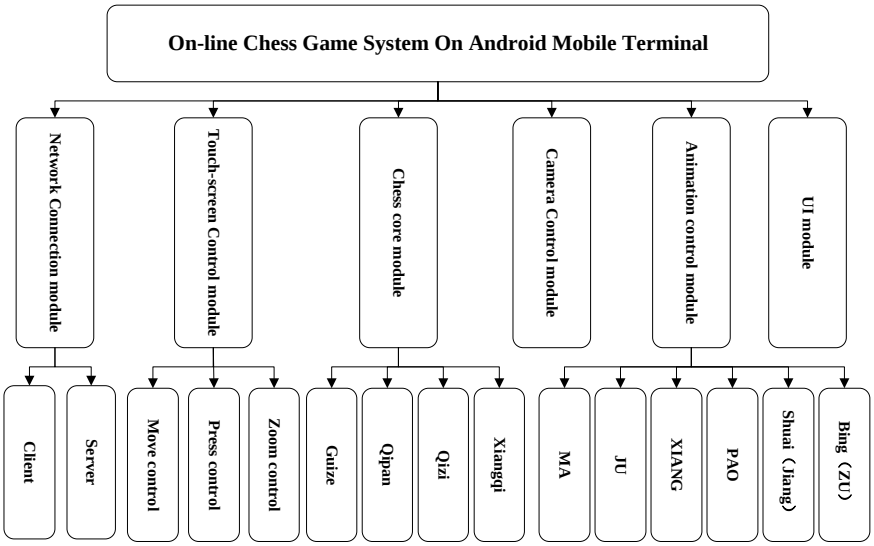

Fig. 1. System structure.

UI module controls the interface of the game and involves the most intuitive part of user experience. It includes twenty-one components like UIAnchor, UIAtlas, UICamera, UIFont and UILabel which respectively control the interactive display of map, camera, input textbox, label, font and other details of the game.

Animation control module controls each model to respond to animation and connects the pictures with core Data Bridge. Due to the feature of Unity3D engine, different response codes need to be set for animation of each kind of character (involving different duration time, distance and speed of movement) in order to realize smooth and coherent pictures. Therefore, there are six sub-modules under animation module, which separately control dynamic state of bing (zu), shuai (jiang), pao, xiang, che and ma (names of chess pieces) such as moving, waiting and attacking, to realize synchronization of pictures and data.

Chess core system module controls the core data of the game and has four sub-modules: XiangQi (chess), QiPan (chessboard), QiZi (chess piece) and GuiZe (rule), which are used to record game data, set game rules, control game proceeding and judge the game result. Optimization algorithm is adopted to reduce the storage space requirement to the lowest.

Camera control module controls the movement of camera to switch the view of the game (the change of view while switching player in control), so as to make the game run naturally and smoothly.

Touch-screen control module is mainly responsible for the game performance on touch-screen mobile terminals. The players can control the game through different gestures. For example, the gesture that two to five fingers zoom out or in at the same time can realize zooming function. This game system follows the principle of gesture interactive design on touch-screen mobile terminals, thus displaying good user experience.

Network connection module is taken to make two mobile terminals to accomplish the communication link and data transmission through WLAN, so as to realize on-line VS play function. Based on the Network View component and Network script class of Unity3D, two Android devices will be connected through Wi-Fi in the LAN to transmit and share data, thus bringing out the on-line VS play effect. This paper will focus on the principle and operating mechanism of this module.

3. Physical Effects and Action Simulation of Natural Scenes

The scene of the game system simulates natural scenes. The simulation technology of physical effects is mainly applied to the simulation of wave, gravity, glass material, props, clothes material and other elements of a game [22]. In this game, the Shader function of Unity3D plays a critical role. In the Unity3D engine, all the graphic plotting needs to use Shader.

3.1. Principle Analysis of Shader and Illumination Model in Unity3D

From the aspect of pipeline, there are two kinds of pipelines: fixed rendering pipeline and programmable rendering pipeline [23], [24].

Fixed rendering pipeline belongs to Geometry and Lighting pipeline (standard Transforming and Lighting pipeline or T&L pipeline for short) with fixed and limited function. It can manipulate properties such as world, shadow and visual conversion, as well as the mixture of texture and fixed lighting. It can operate properly on most of the video cards.

Programmable rendering pipeline, as its name implies, means that some links of the pipeline can be programmed and unnecessary to apply fixed function as fixed rendering pipeline. Instead, parameters can be set to control pipeline and pixel operations and vertex operations can also be programmed.

Three types of Shader in Untiy3D: Fixed Function Shader is part of fixed rendering pipeline and adopts ShaderLab language (grammar similar to FX files of Microsoft or CgFX of NVIDIA), mainly applied to Fallback when advanced Shader cannot display on the obsolete video card.

Vertex and Fragment Shader, adopting CG/HLSL grammar, is a powerful type of Shader and a part of programmable rendering pipeline.

Surface Shader, as a default Shader of Unity3D, also adopts CG/HLSL grammar and uses lighting model prefabricated by Unity to do lighting operations.

Shader of Unity3D is based on several types of lighting model, among which Lambert lighting model and BlinnPhong lighting model are the most common in use [25].

Lambert diffuse reflection model is a type of empirical model which is mainly applied to simulate the illumination of objects with rough surface. This model assumes that the surface of an object is an ideal diffusely reflecting surface (also called Lambert Reflectance). Additionally, there are two types of light in the scene: ambient light and directional light. Do separate calculation of the illumination of two types of light on the rough surface of an object, plus the results and get the number of light intensity after reflection.

The formula for calculating ambient light:

$$\mathrm{I\_ambdiff} = k\_d \bullet l\_a$$

where k_d is the reflection coefficient (provided by the code writer in host) and l_a is the intensity of ambient light, the variable of float3.

The formula for calculating directional light:

$$I\_ldiff = K\_d \bullet I\_l \bullet (N \bullet L)$$

where I_l is the intensity of directional light, the variable of float3, L is the unit normal vector of incoming light and N is the unit normal vector of vertex.

The intensity of light after reflection:

$$I\_diff = K\_d \bullet I\_a + K\_d \bullet I\_l \bullet (N \bullet L)$$

Since the ambient light is the same regardless of the direction, there is no need to calculate its direction. Nevertheless the direction of directional light needs to be calculated.

BlinnPhong lighting model is a modified lighting model of BlinnPhong model. The illumination of fixed pipeline has been calculated on vertex and then the pixel values between vertices can be interpolated. The specular exponent is added and power is adopted to control the specular lighting to smooth the specular edges. The Specular level is in Max is the numerical value of power.

$$I = Il \bullet W(x) \bullet \cos^{n}(x)$$

To the reflected mirror properties in the scene, the parameters of Shader are set as follow:

```
Properties {
_Color ("Main Color", Color) = (1,1,1,1);
_SpecColor ("Specular Color", Color) = (0.5, 0.5, 0.5, 1);
_Shininess ("Shininess", Range (0.01, 1)) = 0.078125;
_ReflectColor ("Reflection Color", Color) = (1,1,1,0.5);
_MainTex ("Base (RGB) Gloss (A)", 2D) = "white" {};
_Cube ("Reflection Cubemap", Cube) = "_Skybox" { TexGenCubeReflect }
}
SubShader {
Cull Off
LOD 300
Tags{
"Queue"="Transparent";
"IgnoreProjector"="True";
"RenderType" ="Transparent";
}
}
```

In the shader, we use the Blinn Phong model for rendering and set the corresponding values for several key parameters, such as the basic color, specular color, shininess, cubemap and SubShader.

Table 1. Time for Game Roles to Achieve Best Performance

	Shi	Xiang	Ma	Pao	Che	Bing	Shuai
Waiting	1.5s	1.5s	1.2s	1.0s	1.0s	1.5s	1.5s
Attacking	1.0s	1.0s	1.2s	1.4s	0.7s	1.0s	1.0s

3.2. The Implementation of Action Simulation

Based on the fact that animation of different game roles have different playing speed and time, in order to make animation effect to be more authentic and smooth, the author attempts to set wait/play time for the animation of different game roles. After repeated tests, the time table of the animation to bring out the best performance is as shown in the Table 1.

The inheritance structure of movement script is shown in Fig. 2.

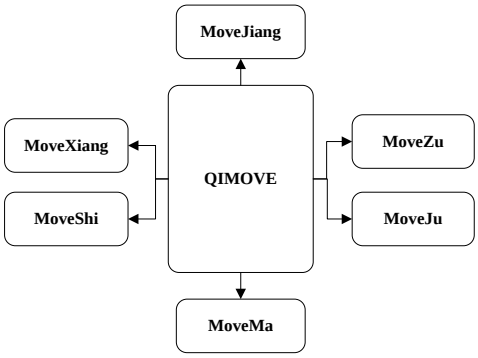

Fig. 2. Structure of movement script.

QiMove class is the general class to control the animation and MoveXiang, MoveZu, MoveJiang,

MoveMa, MoveShi, MoveJu all respectively inherit from QiMove class and rewrite the approaches of Qimove. Here is the structure of QiMove:

```
public class QiMove: MonoBehaviour {
public virtual void ChoosePlayer(){};
public virtual void Cancel(){};
public virtual void Move(Vector3 desPosition){};
public virtual void Attack(Vector3 desPosition){};
public virtual void AttackMove(Vector3 desPosition){};
}
```

Take Xiang () as an example and use the animation file prefabricated. The approach of rewriting Move () can be represented as follows:

```
public override void Move (Vector3 desPosition)
{
//inherit QiMove class and rewrite the moving //animation of “Xiang”.
Vector3 targetDir = desPosition - gameObject.transform.position;
Vector3 forward = Vector3 .forward ;
angle_P = Vector3.Angle(targetDir, forward);
if(desPosition.x<gameObj.transform.position.x)
{
angle_P = -angle_P ;
}
this.FindObject ();
gameObj.animation.CrossFade("Xiang_move")
dummyPos = desPosition;
dummyMove =true;
}
```

The above approach illustrates how to move the game object, create and cancel dummy, play audio and animation, and update the data. Similar approach can be applied to attacking movement and therefore

unnecessary details are not given here.

4. Core Algorithm of Qi class

Core algorithm consists of data storage of Qi class, algorithm and the deployment of chessboard. The structure of core algorithm is shown as Fig. 3.

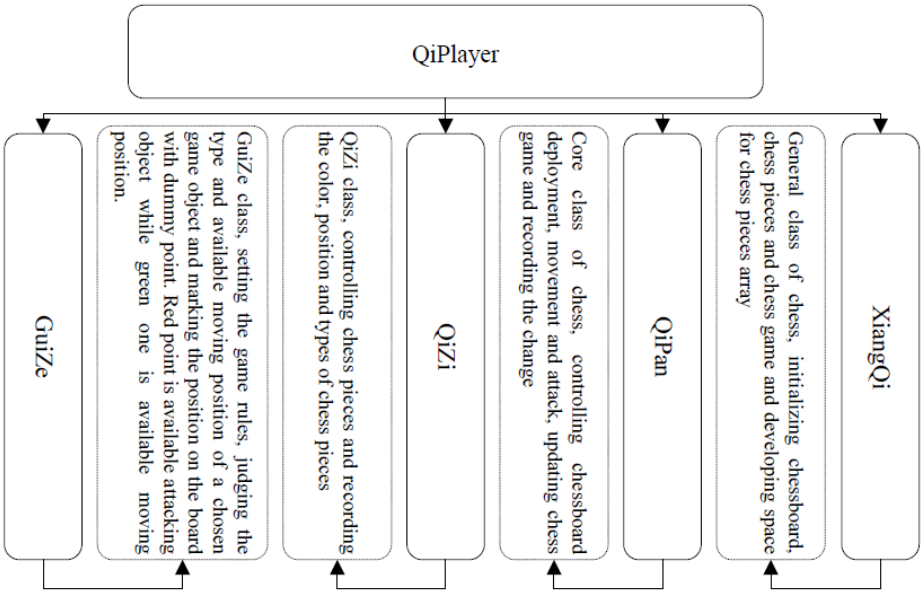

Fig. 3. Structure of algorithm.

QiPlayer is put in Unity3D as an object and four components are added here which are XiangQi, QiPan, QiZi and GuiZe. The functions of these four classes are shown in Fig. 3.

The invoking relationship between these four classes is shown as Fig. 4 (the one at which the arrow is pointing is the class being invoked)

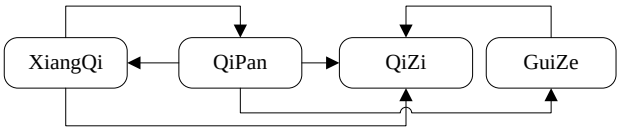

Fig. 4. Invoking relationship between Qi classes.

The sentence to create dummy is as followed:

Instantiate

(QiPan.dummy1 ,pos ,QiPan.dummy1 .transform .rotation);

The first parameter corresponds to the Object of Unity3D, which is the entity of the dummy; the second parameter is the position coordinates of the place creating the dummy; the third parameter is the rotation degree of creating the dummy.

5. The Implementation of Network Module

The network function of Unity3D is originally designed for multiplayer game [26]. It provides online VS game in Master Server mode and also supports C/S connection in LAN defined by users.

Considering the features of the game, the author designs the C/S on-line mode which is created by one player but connected by two. The first player creates a server and the second sends connecting request to the first one. Once the connection is established, the first player will act as the Serve while the second one will exist as Client. However, through the writing of network script of the game which guarantees two-way transmission of data and disconnected control of network, both players of the game seem to be equal from the perspective of users.

5.1. Relevant Scenes

The network module involves three game scenes which are Menu, Two and JZX. The switching relationship is as shown in Fig. 5.

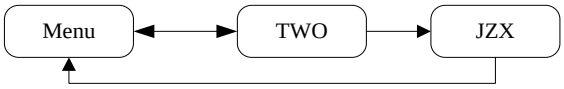

Fig. 5. Switching relationship between scenes.

In the scene Menu, when user chooses online mode, LoadLevelOnClick script component will be used to load scene Two.

```
void OnClick ()
{
    if (!string.IsNullOrEmpty(levelName))
    {
        Application.LoadLevel(levelName);
    }
}
```

Establish the network connection in scene Two and load scene JZX. If the connection hasn't been established, the user can choose to return to scene Menu. Scene JZX is the actual playing interface, in which data transmission and pictures synchronization of both players will be achieved.

5.2. Network Connection

Before building network environment, both players must be in the same LAN. The network can be connected by Wi-Fi or self-developing AP hot point. To build a network connection of Unity, operations will be determined by current network state, which will be obtained by Network.peerType. There are four kinds of state: Disconnected, Connecting, Server, Client, representing disconnected, connecting, operating as server and operating as client. While entering scene Two, the network state will be disconnected. At this moment, the user can invoke Application.LoadLevel by clicking "return" and return to scene Menu. The game distinguishes the server and client through the color of chess pieces. If the user clicks the red avatar, Network.InitializeServer in NetworkServer script will be invoked to initialize the server and the network state will turn to Server.

In the Update function which is invoked in every frame [23], once it is determined as Server, it will show its IP address and wait for the client's connection. When a client connects, scene JZX will be loaded.

```
void Update () {
if(Network.peerType==NetworkPeerType.Server){
    lable.text = Network.player.ipAddress;
    int length = Network.connections.Length;
    if(Network.connections.Length==1)
        Application.LoadLevel(levelName);
```

```
    }
}
```

Blue avatar represents client. If the user types in the IP address of server in the textbox and clicks GO button, Network.Connect function in Network_Client script will send connecting request to the IP which will be judged by the Update function. Once the connection is established, the network state will turn to Client and scene JZX will be loaded. The image of client typing in server's IP address is shown in Fig. 7.

Fig. 6. Game connecting image.

Fig. 7. Client typing in server's IP.

Now the network connection of Unity3D has been built, followed by users' entering the game and data transmission. It is worth noting that after the network connection is built, it will keep the connecting state until Network.Disconnect() function is invoked to disconnect the connection. The access for disconnecting is set in the scene JZX.

5.3. Network Data Transmission

All the actual users' operations occur in the scene JZX, and therefore a large number of data transmissions happen here. There are two kinds of mechanism of Unity3D's network data transmission: State Synchronization and Remote Procedure Calls (PRC). For State Synchronization mechanism, though it does not need the writing of script, it only supports one-way data transmission from client to server. However, both players involved in the game are logically in the equal position, which calls for the implementation of two-way synchronization. Therefore, PRC is adopted by this game, as a reliable, well-organized and more economical communication mode with free transmitting direction. Regardless of mechanism, the process of all the network data transmission in Unity3D begins with adding Network View component to the game object which includes the data to be synchronized.

In the game, the data needs to be transmitted includes the information of start and destination position of the moving chess pieces, rotation timing of both players and game result. This game is not a real-time game

which requires high-quality synchronization, so there is no need to do continuous synchronization of the state of the objects involved. Hence after the game object of one player changes, this one-off game event will be informed only by executing a PRC function on the same game object of the other player.

PRC is a function claiming to be [PRC] in the script which is attached to a game object including Network View component. It can be invoked by function of any other script components in the game object.

The game object QiPlayer attaches Qipan script, in which PRC function is claimed.

[RPC] void qipanrpc(string itsname, intstarti, int startj, Vector3 post, intctag)

In Qipan script, there are two accesses for users' touch-screen operations. These two accesses will process based on users' operations and camera_tag and respectively invoke qipanrpc function to transmit network data.

Firstly the identity of the player needs to be determined and the game default is that server controls red chess pieces while client controls blue ones. Then it comes to the moving logic, refreshing the chess pieces' location and processing animation according to users' operations. The last step is to invoke RPC:

```
void FingerGestures_OnFingerTap( intfingerIndex, Vector2 fingerPos, inttapCount )
{
//the user chooses a chess piece to move and attack
……
if ( IsConnect.isConneted ||
(!IsConnect .isConneted&&Network.isServer&&QPlayer=="RPlayer") ||
(!IsConnect .isConneted&&Network.isClient&&QPlayer=="BPlayer"))
{
//the network is connected and server //controls red pieces
……
}

if (!IsConnect .isConneted)
networkView.RPC("qipanrpc",RPCMode.Others, rpcname, startI, startJ, obj.transform.position, camera_tag);
}
……
}
```

There is no need to identify the user for canceling operation, but RPC function need to be invoked to inform the other player of the game object canceled.

```
void FingerGestures_OnFingerLongPress (intfingerIndex, Vector2 fingerPos )
{
//users' long-press on the screen for canceling //operation
if(camera_tag==0)
{
camera_tag = 2;
//camera_tag 0 represents the chess piece being //chosen while 2 represents ones not being //chosen
qiMove =
QiPan.player.GetComponent<QiMove>();
qiMove .Cancel();
player = null;
if( !IsConnect .isConneted )
```

networkView.RPC("qipanrpc", RPCMode.Others,rpcname, startI, startJ, obj.transform.position, camera_tag);

}

}

Network View is the Network View component attached to game object and its subordinate RPC is the core function of remote invocation. The first parameter is the name of RPC function which is to be invoked and already claimed. The second parameter represents receiving object of RPC invocation. Others means sending to all the others except the sender itself. The rest of the parameters are the ones which need to be introduced for RPC invocation. The network data transmission here includes the name of the operating game object, rpcname, the start position of chess pieces, startI and startJ, the destination position of chess pieces, obj.transform.Position and the tag of game object being chosen, camera_tag. A one-to-one correspondence is necessary here between the types and orders of all these parameters and those of the parameters of the qipanrpc function invoked.

When the program executes function networkView. Rpc, the receiving objects of RPC, at its own end, have already executed appointed RPC function "qipanrpc".

```
void qipanrpc(string itsname, intstarti, int startj, Vector3 post, intctag)
{
if(ctag ==2)
{
qiMove =
QiPan.player.GetComponent<QiMove>();
qiMove .Cancel();
player = null;
   }
   else {
// sending end of RPC executes function //FingerGestures_OnFingerLongPress
obj = GameObject.Find(itsname);
……
}
}
```

Fig. 8. Picture from red party after synchronization.

If the sending end of RPC cancels chosen object, the function FingerGestures_OnFingerLongPress will be executed. According to transmitted camera_tag (ctag of receiving end), the receiving end of RPC will be

informed to execute similar code of function Finger Gestures_On Finger Long Press. If the sending end of RPC chooses a chess piece and makes it move or attack, function FingerGestures_OnFingerTap will be executed here. At the receiving end of RPC, firstly GameObject. Find approach will be used to find the corresponding game object according to the parameters transmitted obj. name (itsname) and then code similar to function FingerGestures_OnFingerTap will be executed according to other parameters.Through the above invoking process of PRC function, the information about the color of chess pieces and the position of game objects will be instantly transmitted to client or server, thus achieving the picture synchronization effect from the perspective of users. The picture synchronization effect is shown in Fig. 8.

Fig. 9. Connection established, game starts.

6. Conclusion

This paper, from aspects such as system structure, game roles, environment simulation, animation implementation and operating mechanics, conducts a detailed discussion about the self-developing 3D chess game. Then combined with Shader function of Unity3D, illumination principle is elaborated and simulation of system environment and animation is realized. Finally, the author gives a detailed introduction about how to establish a network to carry out online VS play function in LAN through Unity network module based on Wi-Fi connection. The design sketch of online play effect is shown in Fig. 9.

Acknowledgements

This research was supported by The Fundamental Research Funds for the Central Universities (RW2014-16). We also thank anonymous referees and the editor-in-chief.

References

[1] Shang, H., & Zheng, Y. G. (2009). Empirical research on the present situation of online game industry. *Reformation and Strategy, 25(1)*, 166-169.

[2] Gao, X. l. (2006). SWOT anylysis on the online game industry in China. *Huazhong Normal University Journal of Postgraduates, 13(2)*, 145-148.

[3] Gao, X., Zheng, Z., & Quan, Y. (2013). Research and design of 3D game engine. *Computer Systems and*

Applications, 22(8), 29-33.

[4] Liu, Y. W., Zhang, Y., & Ye, X. Z. (2006). Architecture of 3D game renderer and its technologies. *Application Research of Computers, (8)*, 45-51.

[5] Sun, X. P., Feng, J. J., Shao, Y. A., & Li, C. F. (2010). Application of UML in game system analysis and design. *Computer Engineering and Applications, 46(13)*,70-72.

[6] Xie, T., Tillmann, N., & De, H. J. (2013). Educational software engineering: Where software engineering, education, and gaming meet. *Proceedings of the 2013 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change, GAS 2013 - Proceedings* (pp. 36-39).

[7] Luo, B., Qin, H. Y., & Huang, C. X. (2013). The generation of a three-dimensional aggregate model for concrete based on unity3D. *Advanced Materials Research,* 3196-3199.

[8] Pickersgill, S. (2007). Unreal studio: Game engine software in the architectural design studio. *Proceedings of the 12th International CAAD Futures Conference on Computer-Aided Architectural Design Futures* (pp. 197-210).

[9] Sun, N. l., & Gu, Z. X. (2013). The research on the interface between unreal engine 3's particle system and external program. *Proceedings of the 2013 7th International Conference on Image and Graphics* (pp. 806-810).

[10] Gregory, S. C. (2013). Implications of a multidisciplinary students' computer game design project. *Journal of Computer*, *8 (7)* , 1836-1840.

[11] Nie, D. X., & Li, Y. B. (2012). On the philosophical thought of Chinese chess. *Journal of Zunyi Normal College*, *14(4)*, 11-12.

[12] Zhang, H. A. (2013). Image-numerological Culture of the Yi learning reflected in Chinese chess. *Journal of Chengdu Sport University*, *39(6)* , 53-57.

[13] Wang, X. P., Wang, J., Xu, X. H., & Zheng, X. Y. (2007). A comparative analysis between chess and Chinese chess. *Journal of Chongqing Institute of Technology*, *21(1)*, 71-76.

[14] Yuan, J. P., Li, Y., Hao, Z. Y., & Wang, Y. J. (2012). Multi-party dialogue games for dialectical argumentation. *Journal of Computer*, *7(10)*, 2564-2571.

[15] Hao, Z. X., Lei, X., Lin, F. B., & Qu, X. L. (2012). MD3 model loading in game. *Journal of Computer, 7(2)*, 521-527.

[16] Liu, H. (2013). Chinese chess 3D-digital modeling character design. *Art and Design*, 106-108.

[17] Leandro, M. N., Eduardo, S. D. Almeidal, & Silvio, R. D. L. M. (2008). A case study in software product lines-the case of the mobile game domain. *Proceedings of the 34th Euromicro Conference Software Engineering and Advanced Applications* (pp. 43-50).

[18] Cho, H. J., & Yang, J. S. (2008). Architecture patterns for mobile games product lines. *Proceedings of the 10th International Conference on Advanced Communication Technology* (pp. 118-122).

[19] Xu, C. W. (2008). A software framework for online mobile games. *Proceedings of the International Conference on Computer Science and Software Engineering* (pp. 558-561).

[20] Zhao, H. Z., Wang, W., & Ming, Y. L. (2011). Design and optimization of a web 3D massively multiplayer online game engine. *Proceedings of Tudy on Link-Level Simulation in Multi-Cell lte Downlink Syste* (pp. 296- 300).

[21] lu, J. T. (2013). Teaching practice of 3Dmax course for animated ads. *Proceedings of the International Conference on Education Technology and Information System* (pp. 410-415).

[22] Martin, R. (2009). Video-based running water animation in Chinese painting style. *Science in China (Series F: Information Sciences)*, 162-171.

[23] Su, Y., Chen, C. Y., & Guo, S. L. (2006). Computing technology of lighting based on vertex shader.

Computer Technology and Development, *16(12)*, 58-60.
[24] Zhu, T. H., Liu, X. H., & Wu, E. (2002). Per-pixel lighting. *Journal of Computer Aided Design and Computer Graphics*, *14(9)*, 861-865.
[25] Chen, S., Shum, H. Y., & De, M. A. S. (2000). A survey of image-based illumination model. *Chinese Journal of Computers*, *23(12)*, 1261-1269.
[26] Zhu, H. J. (2012). Virtual roaming system based on unity3d. *Computer Systems & Applications*, *21(10)*, 36-39.

Dayan Shangguan was born in 1979. He received his master's degree in 2009 from Peking University. Now he is a lecturer in Beijing Forestry University. His research interests include VR technology and computer-supported cooperative work, etc.

Xintong Li was born in 1991. She received her bachelor's degree in 2014 from Beijing Forestry University. Her research interests include VR technology and computer-supported cooperative Work, etc.