(1) Workshop:

Workshop on Databases and Software Engineering.

(2) Title:

The Formalization and Design of the General Play-on-table Game System.

(3) Abstract:

For most traditional board/card games, such as Chess, Chinese Chess, Go, Chinese Mahjong, Hearts, Bridge, etc., that people just sit to play around a table, share the same playing model. In this paper, we introduce and formalize the definition of this model. These games following the model are named as play-on-table (abbr. POT) games.

In this paper, we propose a game system to allow players to play all the POT games. By "play", we mean that we can simulate all POT game plays in this game system. We also implement such a game system on top of the CYC game server over Internet.

(4) About Authors:

Names: I-Chen Wu and J. J. Hsu

Affiliation and postal address: Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan

E-mail address: icwu@csie.nctu.edu.tw and jjshie@csie.nctu.edu.tw.

TEL: +886-3-5731855. FAX: +886-3-5733777.

(5) Contact Person:

I-Chen Wu

(6) List of Keywords:

Internet, play-on-table games, game servers.

Workshop on Databases and Software Engineering

The Formalization and Design of the General Play-on-table Game System

I-Chen Wu and J. J. Hsu

Department of Computer Science and Information Engineering

National Chiao Tung University, Hsinchu, Taiwan

icwu@csie.nctu.edu.tw and jjshie@csie.nctu.edu.tw

Contact person: I-Chen Wu

TEL: +886-3-5731855 FAX: +886-3-5733777

Abstract

For most traditional board/card games, such as Chess, Chinese Chess, Go, Chinese Mahjong, Hearts, Bridge, etc., that people just sit to play around a table, share the same playing model. In this paper, we introduce and formalize the definition of this model. These games following the model are named as play-on-table (abbr. POT) games.

In this paper, we propose a game system to allow players to play all the POT games. By "play", we mean that we can simulate all POT game plays in this game system. We also implement such a game system on top of the CYC game server over Internet.

Keywords: Internet, play-on-table games, game servers.

1. Introduction

With the rapid development and the fast-growing user base of Internet, it becomes significant for users to communicate with others via Internet. Thus, this forms a virtual community for groups of people with the same interests. Gaming is one of the most applications in the Internet virtual community.

In this paper, we focus on those traditional board/card games, such as Chess, Chinese Chess, Go, Chinese Mahjong, Hearts, Bridge, etc., that people just sit to play around a table. Since these games share the same playing model, we introduce and formalize the definition of this model. And, these games following the model are named as play-on-table (abbr. POT) games in this paper.

In order to allow POT games to be played over Internet, there were two main solutions. The first solution is to develop a game server for each single game, such as Free Internet Chess Server (FICS) [1], Internet Go Server (IGS) [2], and QKMJ [5]. The drawback of this solution is: Each server can only support one game. When the programmers decide to write the game server for one POT game, they still need to develop from scratch (e.g., at least need to implement the room management and the membership management.).

The second solution is to develop POT game servers for a series of games, such as Acer Game Zone [7], Yahoo! Games[6], the CYC game server [3,4]. For example, Yahoo! Games include the following traditional board games and card games: Backgammon, Checkers, Chess, Go, Reversi, Blackjack, Bridge, Hearts, Poker, etc. Although the second solution can provide players with many POT games, the problem is still: No matter how many games are

supported, this solution does not cover *all* the POT games.

In this paper, we propose to design a game system to allow players to play all the POT games. By "play", we mean that we can simulate all POT game plays in the game system. We also implement such a game system on top of the CYC game server over Internet.

In Section 2, we formally define the model of POT games and prove that a game system can be derived to simulate all the POT games. In Section 3, we implement such a game system. This system is implemented on top of the CYC game server over Internet. In Section 4, we give a discussion.

2. Play-on-table (POT) Game Model

In this section, we will formally define POT games. First, for simplicity, we define the simplified POT (abbr. SPOT) games for those games where all objects are put in public on the table in Section 2.1. Second, in Section 2.2, we define the POT games to allow players to hold game objects in their own private areas. In addition, we will prove that some game systems can be designed to simulate all the POT games. Namely, these game systems allow players to play all the POT games.

2.1. Simplified POT (SPOT) Games

A SPOT game is $G = (A, O, S, M)$ as defined as follows.

- A is a two-dimensional *game area* representing the table.
- O is a finite set of *game objects*. Namely, $O = \{o_1, o_2, o_3, ...o_n\}$, where each object o_i has

m_i faces, $\Phi_i = \{\varphi_{i0}, \varphi_{i1}, \varphi_{i2}, ..., \varphi_{i,mi}\}$. Now, from A and O, we can derive all the possible *table states* $\Gamma(A,O)$ as follows:

$\Gamma(A,O) = O_1 \times O_2 \times ... \times O_n$, where $O_i = A \times Z \times \Phi_i$ and Z is the set of z-order indices (we can simply use the set of real numbers). O_i is said to be the set of *states of object* o_i. For each table state $q = ((p_1, z_1, \varphi_1), (p_2, z_2, \varphi_2), ..., (p_g, z_g, \varphi_g))$ and for each object o_i, the object state (p_i, z_i, φ_i) indicates that this object o_i turns to the face φ_i at position p_i with z-order z_i (the distance to the viewers and the larger the closer). One key for POT games is that once games are set all objects will not disappear on the table.

- S is a set of legal *game states*. Namely, $S = \{start, end\} + S^{-}$, where S^{-} is a subset of table states $\Gamma(A,O)$. For a game, players start from the state *{start}* and end at the state *{end}*.
- M is a set of legal *game operations*, representing the game rule. Namely, $M = S \times S$, where each legal operation (s, s') indicates that this operation changes the game state s to another s'. Note that one game state can be changed to several different game states.

A sequence of game states $(q_1, q_2, q_3, ...q_g)$ is called a *game play segment* in the above game G, if each pair of states (q_i, q_{i+1}) in the sequence is a legal game operation in M. A sequence of game states $(q_1, q_2, q_3, ...q_g)$ is called a *game play* in G, if the sequence is a game play segment and $q_1 \in \{start\}$ and $q_g \in \{end\}$.

For each game state, its *game appearance* is the image that the players see on the table. Namely, for the two game states *{start}* and *{end}*, the appearances are empty images; for a game state (or table state) $q = ((p_1, z_1, \varphi_1), (p_2, z_2, \varphi_2), ..., (p_g, z_g, \varphi_g))$, the appearance is the image canvas painted in the following steps:

- Sort the objects according to their Z values.

- For each sorted object, say o_i, starting from the object farthest from the viewers or with the smallest z-order, paint its face φ_i, at position p_i on the image canvas.

Let us consider the game Tic-tac-toe G_T as an example as follows.

- $G_T = (A_T, O_T, S_T, M_T)$.
- A_T is a table with 3 × 3 grids (at position $p_{11}, p_{12}, p_{13}, p_{21}, \ldots, p_{33}$) plus a place (say at position p_{00}) storing unused objects.
- O_T has five white pieces and four black pieces, where each piece is put at one of the above 10 positions.

  As above, the set of all the table states $\Gamma(A_T, O_T)$ is $O_1 \times O_2 \times \ldots \times O_n$, where $O_i = A_T \times Z \times \Phi$. For simplicity, let the first 5 objects be white pieces and the rest 4 objects be black ones, and let all objects have the same Z value, say 0. (Note that if we want to make appearance always the same we can let all the white pieces have Z value 1 and all the black pieces have Z value 0.) Thus, since Z has only one value and has also one value, $\Gamma(A_T, O_T)$ can be simplified as $A_T \times A_T \times \ldots \times A_T = A_T^9$.
- S_T^- is a set of game states, $\{p_{00}, p_{11}, p_{12}, p_{13}, p_{21}, \ldots, p_{33}\}^9$;
- M_T is a set of legal game operations as follows:
  - The initial operations: $\{(\{start\}, q) \mid q \in S_T^-$, all objects of q must be at $p_{00}\}$.
  - Each legal move operation: $\{(q_i, q_j) \mid q_i, q_j \in S_T^-$, q_i is the same as q_j except for that one white (black) piece in q_i is moved from p_{00} to a grid which is empty in q_i, if q_i is in the white-turn (black-turn)$\}$. We can decide q_i is in the white-turn (black-turn) by checking whether the number of white (black) pieces at p_{00} is higher than the number of black (white) pieces at p_{00}.
  - The end operations: $\{(q, \{end\}) \mid q \in S_T^-$, each grid except p_{00} has one and only one object$\}$.

A game play G_T is illustrated in Figure 1.

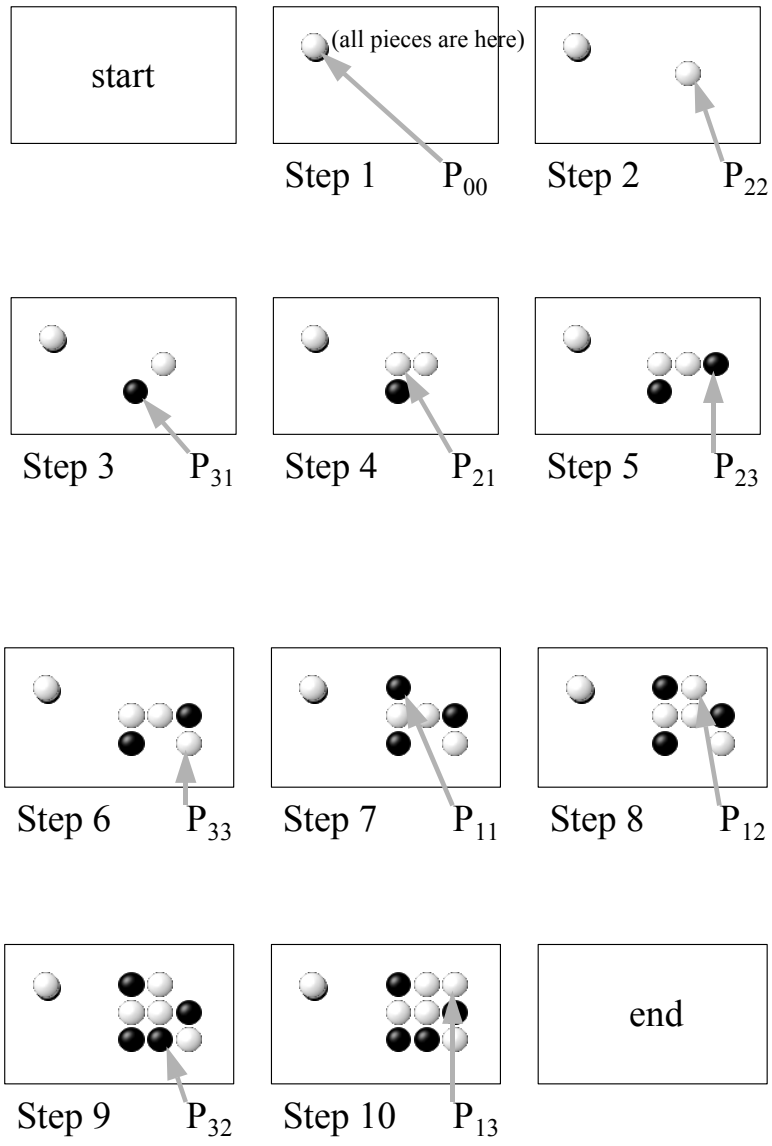

Figure 1. A game play of Tic-tac-toe.

Consider two SPOT games $G = (A, O, S, M)$ and $G' = (A', O', S', M')$. A game play, $(q_1, q_2, q_3, \ldots, q_g)$, in G is called a *partial game play* of another game play, $(q'_1, q'_2, q'_3, \ldots, q'_{g'})$ in G', if $q_i = q'_{m_i}$ for each i where $1 \le m_1 < m_2 < \ldots < m_g \le g'$.

If we want a game play ρ to simulate another ρ', we only need the appearances of the

game play ρ' to be the same as those of some partial game play of ρ'. More specifically, a game play, $(q_1, q_2, q_3, \ldots, q_g)$, in G can be *simulated* by another game play, $(q'_1, q'_2, q'_3, \ldots, q'_{g'})$ in G', if q_i has the same appearance as q'_{m_i} for each i, where $1 \leq m_1 < m_2 < \ldots < m_g \leq g'$.

A SPOT game G can be *simulated* by another SPOT game G', if for each game play ρ in G there exists some game play in G' that can simulate the game play ρ.

A *SPOT game system* is defined to be a set of SPOT games. A SPOT game system is *general*, if all the SPOT games can be simulated by a game in the game system.

Now, we will design a general SPOT game system that only takes objects as parameters. Let the general SPOT game system be $\gamma(O) = (A_\gamma(O), O, S_\gamma(O), M_\gamma(O))$ as follows.

- $A_\gamma(O)$ is the whole two-dimensional space, R^2, since all game areas A belong to the whole two-dimensional space.
- $S_\gamma(O) = \{start, end\} + \Gamma(R^2, O)$.
- $M_\gamma(O)$ is the union of the following three sets:
  - The set of initial game operations: $\{(\{start\}, q) \mid q \in \Gamma(R^2, O)\}$.
  - The set of change-object operations: $\{(q, q') \mid q, q' \in \Gamma(R^2, O)$; and the game state q is the same as q' except for that only one object may have different object states in both q and $q'\}$.
  - The set of final game operations: $\{(q, \{end\}) \mid q \in \Gamma(R^2, O)\}$.

Theorem 1 shows a property that the above game system $\gamma(O)$ is general.

Theorem 1. The above game system $\gamma(O)$ is general for all SPOT games.

Proof. Consider each SPOT game $G = (A, O, S, M)$. Then, there is a corresponding SPOT game $G' = (A_\gamma(O), O, S_\gamma(O), M_\gamma(O))$ in the game system $\gamma(O)$. Now, it suffices to prove that for each game play ρ in G, there exists a game play ρ' in G' simulating the game play ρ.

Let ρ be a game play, $(\{start\}, q_1, q_2, q_3, \ldots q_g, \{end\})$, in G. First, we construct a sequence of game states, $(\{start\}, q'_1, q'_2, q'_3, \ldots q'_g, \{end\})$, in G' such that all object states in each q'_i are the same as those in q_i. Note that since the area of G' is R^2 covering the whole 2-D space, the Z-order space is the same, and the object set is the same, each object state in game G is also a valid object state in G'.

Due to the initial and final game operations of game G', both operations $(\{start\}, q'_1)$ and $(q'_g, \{end\})$ are legal. For each pair of game states (q'_i, q'_{i+1}), we can insert some game states such that for each pair of neighboring game states there is at most one object whose state is different in both neighboring game states. In other words, each pair of game states is a legal change-object operation. Therefore, the sequence of game states including the inserted game states form a game play in G'. Thus, the game play ρ can be simulated by the game play ρ'. ▌

Furthermore, in $M_\gamma(O)$, each change-object operation is composed of at least one of the following two operations:

- Move-object operation: A move-object operation is a change-object operation, but the faces are not changed.
- Change-face operation: A change-face operation is a change-object operation, but the positions and the z-orders are not changed.

Therefore, let $M'_\gamma(O)$ include move-object operations and change-face operations only and the

game system $\gamma'(O) = (A_\gamma(O), O, S_\gamma(O), M'_\gamma(O))$. Then, the game system $\gamma'(O)$ is still general for all SPOT games.

2.2. POT Games

In the model of SPOT games, we assume that all players will watch everything on the table. However, for the games such as bridge, players can see their own cards that other players cannot see. Therefore, in the model of POT games, we need to define a set of areas, one for all players, called the common area, and every other for one distinct player.

A p-player POT (abbr. p-POT) game is $(A^{(p)}, O, S, M)$, as defined as follows.

- $A^{(p)}$ is a set of 2-D areas, $\{A_0, A_1, A_2, A_3, ..., A_p\}$.
- O is the same as the definition of the object set for SPOT games.

  Similarly, we define table states $\Gamma(A^{(p)}, O) = O_1 \times O_2 \times ... \times O_n$, each $O_i = A^{(p)} \times Z \times \Phi_i$.

  For the position in $A^{(p)}$, we use (n, p) to indicate the position p in area A_n. So, we can use $(n_i, p_i, z_i, \varphi_i)$ to indicate the game state of object o_i.
- S is the subset of $\{start, end\} + \Gamma(A^{(p)}, O)$.
- M is the same as the definition for SPOT games.

In each game state $q = ((n_1, p_1, z_1, \varphi_1), (n_2, p_2, z_2, \varphi_2), ..., (n_g, p_g, z_g, \varphi_g))$, the ith game object is put at position p_i in area A_{n_i} with face φ_i and z-order z_i.

For each game state, its game appearance is the images that players see for all areas. More specifically, for a game state $q = ((n_1, p_1, z_1, \varphi_1), (n_2, p_2, z_2, \varphi_2), \ldots, (n_g, p_g, z_g, \varphi_g))$, the appearance has $p+1$ image canvases and painted in the following steps:

- Sort the objects according to their z-order.

- For each sorted object, say o_j, starting from the object farthest from the viewers (or with the smallest z-order), paint a face φ_j, at position p_j on the n_j-th image canvas.

The appearance for the game state q includes all the appearances of the game state for all players. Obviously, SPOT games can be viewed as a special case of POT games, such as 0-POT games.

An example of a poker game is illustrated in Figure 2. Figure 2(a) and (b) show the appearances before and after moving heart four from the common area A_0 to the private area A_2.

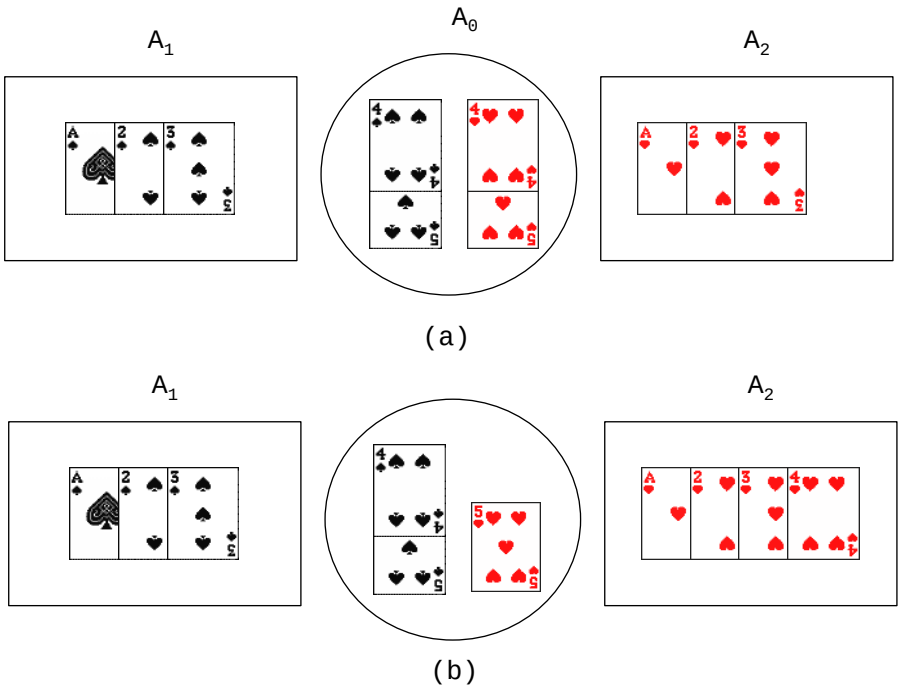

Figure 2. (a) The appearance before moving heart four. (b) The appearance after moving heart four.

No matter how we define the appearances of a game state, it is always true to say that appearances of two game states are the same if all object states of one game state are the same

as those of another. Therefore, we can define that a game play in a *p*-POT game can be simulated by another *p*-POT game in the same way as SPOT games.

A *p*-POT game *G* can be simulated by another *p*-POT game *G'*, if for each game play ρ in *G* there exists some game play in *G'* that can simulate the game play ρ.

A *p*-POT game system is defined to be a set of *p*-POT games. A *p*-POT game system is general, if all *p*-POT games can be simulated by a game play in the game system. A *p*-POT game system is defined to be a set of *p*-POT games (the number of players not necessarily fixed to *p*). A *p*-POT game system is general, if all *p*-POT games can be simulated by a game in the game system.

Now, we will design a general *p*-POT game system that only takes objects as parameters. Let the general *p*-POT game system be $\gamma^{(p)}(O) = (A^{(p)}{}_{\gamma}(O), O, S_{\gamma}(O), M_{\gamma}(O))$ as follows.

- $A^{(p)}{}_{\gamma}(O)$ has *p* areas each of which is the whole two-dimensional space, R^2.
- $S_{\gamma}(O) = \{start, end\} + \Gamma(A^{(p)}{}_{\gamma}(O), O)$.
- $M_{\gamma}(O)$ is the union of the following three sets:
  - The set of initial game operations: $\{(\{start\}, q) \mid q \in \Gamma(A^{(p)}{}_{\gamma}(O), O)\}$.
  - The set of move-object and change-face operations: $\{(q, q') \mid q, q' \in \Gamma(A^{(p)}{}_{\gamma}(O), O)$; and the game state *q* is the same as *q'* except for that only one object may have different object states in location (including the z value) or the face$\}$.
  - The set of final game operations: $\{(q, \{end\}) \mid q \in \Gamma(A^{(p)}{}_{\gamma}(O), O)\}$.

Theorem 2, similar to Theorem 1, shows that the above game system $\gamma^{(p)}(O)$ is general.

Theorem 2. The above game system $\gamma^{(p)}(O)$ is general for all p-POT games.

Proof. Similar to the proof of Theorem 1. ▌

Now, we will design a general POT game system that only takes number p and objects O as parameters. Let the general POT game system be $\gamma(p,O) = (A^{(p)}_{\gamma}(O), O, S_{\gamma}(O), M_{\gamma}(O))$. Thus, we can derive the Theorem 3.

Theorem 3. The above game system $\gamma(p,O)$ is general for all POT games.

Proof. Omitted. ▌

3. The Implementation of the General POT Game System

From Theorem 3 (in the previous section), we know that the POT game system $\gamma(p,O)$ is general, that is, for all POT games G there exists some game G' in $\gamma(p,O)$ that can simulate G. So, in this section, we implement a physical general POT game system based on $\gamma(p,O)$ to support all POT games.

In the game system $\gamma(p,O)$, we need to set up the number of players and all the game objects. So, in our real game system, the players initially set the number of players and the URL location of the package of the game objects, as shown in Figure 3. If the players want to support their own game objects, the players can click on the "design game (1)" button to set their own game objects from a window as shown in Figure 4. This window allows the players to build the set of game objects by inputting objects' names, faces, the initial faces, and the

initial locations (the area index, coordinates x, and y). Then, the user can either save the defined objects or continue to play games (by clicking on the button “done”).

Figure 3. Setting the initial parameters.

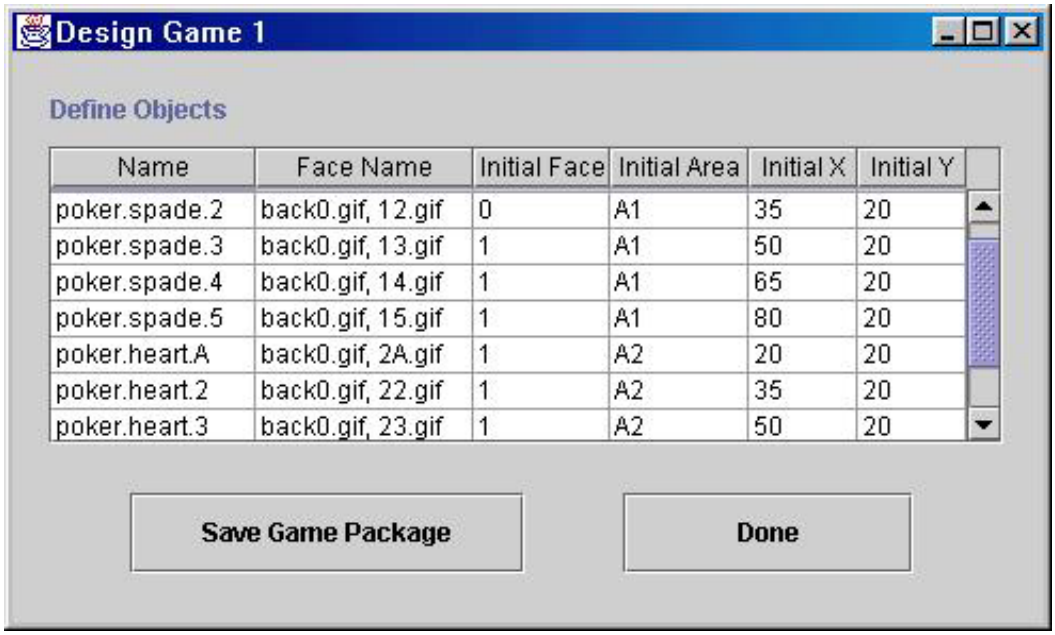

Figure 4. Designing game objects.

Since Theorem 3 shows that it is general enough for the game system to support the move-object and change-face operations, our physical game system supports the two functions in the following ways:

- The move-object operations: The players can use traditional drag-and-drop actions to change location: click on the targeted object, hold it, move it to the destined location, and then drop it. The players can click the right button on the selected object to set the z value.
- The change-face operations: The players can double click on the targeted object to change the face of this object to the next one (in a round-robin manner).

Now, we can see that our physical game system has implemented all the basic elements in the POT game system $\gamma(p,O)$. So, it can be concluded at this point that our physical game system as above has enough power to simulate (or play) all POT games.

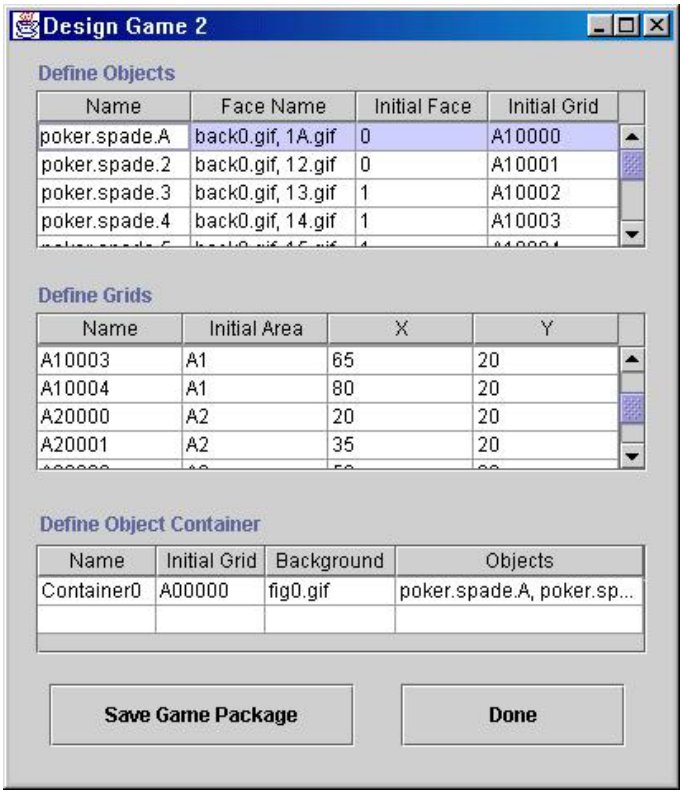

Figure 5. Designing game objects, grids, and game containers.

For real game playing, it is actually helpful to add more features, such as defining grids, supporting game containers, and supporting rotation. If players want to have these features, users can click on the “design game (2)” button in the window of Figure 3. Then, a window is popped as shown in Figure 5 above.

Grids

For most games, such as Chess, Chinese Chess, and Othello, game objects usually stay at some fixed positions. It is even annoying if these game objects are not well aligned. Therefore, it would be useful to allow players to define grids which all the game objects can be located at.

In our physical game system, players can define the name, area index, coordinate x, and y of each grid from the window in Figure 5. For each object, players can specify the initial grid to put. When players move objects, if game objects are not dropped right at these grids, our system will choose the closest grids to put the objects.

Game Containers

For most games, such as most board games, there are some game containers to store game objects. Game objects can be retrieved from or stored into the game containers. Theoretically, we can think these game objects in the containers are located at the same position with different z values. In order for players to grab a game object from the containers, our game system lets the drag-and-drop operation to choose the one in the Z order, the first-in-first-out

order, the first-in-last-out order, and the random order.

Interestingly, for card games, if a stack of cards is viewed as a game container and we set it in the random order, then the stack does card shuffling automatically.

In our physical game system, for each game container, players can specify its name, its background face, its initial grid (location), the game objects allowed to be stored, and the ordering of selecting objects, as shown in Figure 5.

Whiteboard Design and Rotation

Our physical game system allows multiple players to play the game, so our system needs to implement a whiteboard-like mechanism to transmit the game states to all players over the computer network.

If we display the game in the same direction for all players, some players may feel awkward. For example, for Chess, both players see the pieces with the same color (say black) are in the lower side; in other words, the player holding white may still see black pieces in the lower side, contradictory to the normal play feeling.

In order to solve this problem, we assume that all players get together around a table, and we allow players to choose an angle to rotate the table. In addition, it is an option for players to decide whether game objects need to be rotated too. Thus, players can choose the best way to view their games.

The CYC Game Server

Our physical game system described in the previous section is implemented on top of the CYC game server [3,4]. The CYC game server is a framework for game developments. They proposed the multi-client servlet model as shown in Figure 6. Based on this model, game providers only need to write the GUI and game rule parts for games. The game table management and the membership management are handled by the CYC game server framework. Therefore, it is very suitable to demonstrate our game system by leveraging the framework of the CYC game server.

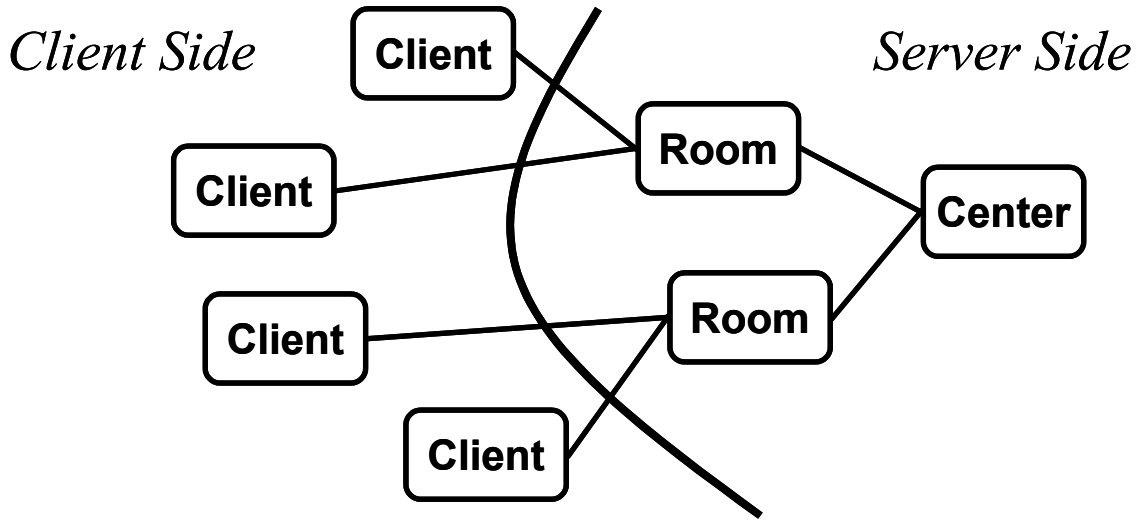

Figure 6. The multi-client game model.

4. Discussion

In this paper, we formally define the model of POT games, which people can just sit around a table to play. Most well known traditional board/card games, such as Mahjong, Chess, Chinese Chess, Go, hearts, bridge, big-2, etc., are all the POT games.

In this paper, we also prove that a game system can be designed to play all POT games in theory. And we also practically implement such a game system on top of the CYC game server over Internet.

Based on the work in this paper, we will continue to investigate whether we can support a game system where players can define some game rules online before playing. We believe that the theoretical model proposed in this paper is helpful for such an investigation.

References

[1] FICS, "Free Internet Chess Server", 1998, available from http://www.freechess.org/.

[2] NKB Inc., Internet Go Service, 1992, available from http://igs.joyjoy.net/.

[3] I-Chen Wu and Cheng-Da Shen, "The Game Developer Guide for the CYC system Version 2", Internal Document, Dec 7, 1997.

[4] I-Chen Wu, "Internal Design Specification for the CYC System", Internal Document, Dec 7, 1997.

[5] S. Y. Wu, "QKMJ 網路麻將", 1997, available from http://www.csie.nctu.edu.tw/~sywu/qkmj.html.

[6] Yahoo!, "Yahoo! Games", 1999, available from http://games.yahoo.com/.

[7] 宏碁戲谷, "Acer Game ZONE 宏碁戲谷", 1999, available from http://www.acergame.com.tw/.