Web-Enabled 3D Game Playing for Looped Knight's Tour

Gregory C.L. Lum and David Y.Y. Yun

Holmes 492, 2540 Dole St., Honolulu, Hawaii 96822 USA
{lumg,dyun}@hawaii.edu

Abstract. This paper elucidates the development of a 3D graphics display environment that facilitates the finding of a closed Loop Knight's Tour (LKT) that uniquely covers each grid in a 3D rectangular box. When LKT is played as a solitaire game in 3D space, it is not only mentally challenging but also difficult for the player to visualize the current or past (occupied) positions and to consider any follow-on possibilities (open grids). These graphic facilities simplify the visualization the global box as occupied and open grids and allow the convenient examination of the sequential chain of knight's moves. Relevant information and valuable relations are computed and displayed to assist the player in choosing the next grid to occupy and closing the ends to form a loop. This graphic game environment is Web enabled via Google's SketchUp. An online community may be developed as users challenge one another by increasingly difficult configurations.

Keywords: knight's tour, closed-loop, 3D grid box, solitaire game, community challenges, growing solved database, Web interaction, Google, SketchUp.

1 Introduction

Just as hard as imagining playing with the Rubik's Cube in an online virtual 3D display, visualizing all the pieces and (lines of) attacks for 3D chess is equally difficult. The fundamental difficulty lies in the visualization of 3D movements on a 2D screen. To achieve online display and enable Web-based playing of even solitaire games, therefore, must rely on motion support (turning, zooming, layering, etc.) in a 3D environment. Computed information such as available positions, attack points, blocking, etc. can be added advantages offered to players via the Web. More difficult aspects in 3D include look-ahead or even anticipating the ending can also be offered by an interactive display environment online. All these essential features (significantly different from those animations for arcade games) are considered and supported to enable an interactive game to be played over the Web. Online players can also post (mutual) challenges and help grow the common solution database for the collective Web community.

1.1 Background

The Knight's Tour (KT) problem originated in the 1700s by an English mathematician named Brook Taylor [8]. He wondered how to cover the 8x8 chess board uniquely by

A. Nijholt, D. Reidsma, and H. Hondorp (Eds.): INTETAIN 2009, LNICST 9, pp. 131–142, 2009.

(exactly 64) knight movements. In 1991, Allen Schwenk, completely solved the KT problem for all 2D rectangular boards. His definition of Knight's Tour is one where the knight visits every square of the board and returns to the original position, hence forming a loop. In order to remove confusion, such closed-ended chain of knight moves will be considered as solving the Loop Knight Tour (LKT) problem. Schwenk [6] dealt with 2D problems only, an m x n chess board, where m is less than n, found that it has a solution unless: (a) m and n are both odd; (b) m = 1, 2, or 4; or (c) m = 3 and n = 4, 6 or 8. He also showed how to combine solutions from boards of smaller sizes. This LKT problem can certainly be extended to 3D and posed as a challenging and interesting solitaire game.

1.2 Motivation

LKT in 3D, unfortunately, cannot be physically realized. Hence, it is forced to use computer based visualization and support. A 2D display, or a sequence of 2D slices, will not be sufficient to clearly represent the LKT snapshots nor any knight's move possibilities among the open and occupied grids in 3D. An enhanced 3D environment is required. This paper demonstrates the design and implementation of a 3D game environment that assists a player with both local/global tracking tools and 3D visualization (with motion) for playing the LKT game. Presently, there is no visual system to make such 3D game playing (including true 3D chess) effective and enjoyable.

1.3 Loop Knight's Tour (LKT) in 3D

The LKT setting can be represented in 3D as a "Game Box" of dimensions l x m x n of $l*m*n$ grids for the knight to move to and from. Each grid is uniquely identified by the coordinate label of (x, y, z). A knight movement is ± 2 and ± 1 in any two directions and order along x, y or z axes. From any (x, y, z) grid in the Game Box there is a maximum of 24 grids that can be reached in one knight movement without getting out of the edges. As each knight movement occupies a grid, it gets a sequential move number, which also is a tally of the total number of grids "toured" so far. [10, 11].

The LKT solution is not completed until each grid of the Game Box has been occupied and the starting and ending positions are the same and a loop tour is accomplished. Before this happens the sequence of movements already completed is a chain of movements with two open ends. Each end can be extended in any order to complete the loop tour and cover every one of the lmn grids. Two chains can be linked together to form a longer chain so long as one end of one chain is a knight's move away from an end of another chain. In fact, since a loop can be "broken" at any link to form a chain, two (adjacent) loops can be broken at chosen links so that one end from each loop can be linked by a knight's move, then a longer joint chain is formed. If those two remaining ends of the joint chain happen also to be a knight's move apart, then a joint loop is found/constructed. Judiciously choosing the links to break from two loops so that two (opposite) pairs of ends are knight movements apart and form a joint loop is the method of "**stitching**" [6] for constructing larger 2D solutions from smaller ones. Such joining of loop solutions can be extended to 3D LKT problems (known as "**knight's crossing**") by stitching together two 2D layers or

two 3D blocks in a similar manner. The simpler form is to stack two LKT loops (maybe the same) in 2D layers one on top of the other. In order to combine the solutions a knight's crossing needs to be found, the easiest of which is at one of the corners. When a knight's crossing is found, the corresponding links (in both loops) need to be broken and the opposite ends reconnected by night's moves to form a larger loop solution, now in 3D [10, 11]. In this way, LKT loops previously found in 2D or 3D can be combined to solve other larger Game Boxes. These loop checking and loop linking are necessary capabilities provided in the LKT game environment to assist the extension of solved boxes or layers into larger LKT solutions.

2 Game Features

Google's SketchUp program provides an online system for composite 3D objects to be accurately represented by 2D screen displays. SketchUp allows the user to control the environment by zooming, panning of an object, view angle or orientation control, and layer removal ("Section Plane") [7]. However, SketchUp needs additional features to properly support the LKT game environment. In order to determine what are required for an online LKT game player, a list of essential support features were developed, from which many 3D modeling tools were considered. SketchUp was selected as the implementation system by meeting most of the requirements of built-in features and its extension capabilities where essential game supports and displays can be created using a combination of Ruby, Java and HTML programming languages [2, 3, 4, 5, 9].

2.1 Two Move Directions

Before the loop solution for the LKT problem can be found, a chain of knight's moves always have two ends, both of which can be extended alternatingly, in a "forward" direction and a "backward" direction. These directions are, of course, arbitrarily labeled, where the ultimate goal is still to connect the ends together and form a loop. In fact, since a loop solution (if exists) can be broken anywhere, the ends of the chain or the starting position matters very little. Rather, they are merely convenient notions (or notations) for the game to continue.

2.2 Game Box

The 3D Game Box is a wire frame transparent grid system of size l x m x n, to allow maximum visibility through the entire game region. Other useful information can be placed within each grid. At the center of an occupied position is a knight piece, which is used to signify that the particular position has already been visited/toured. Next to the knight piece is its move number, which is both the sequence label and the present tally of the number of occupied grids. The move number is also mirrored in the move tables, and allows a quick scanning of the sequence to recognize the path the knight has taken to get to a particular position.

With all of the other features made to be invisible the Game Box would look just as if it was drawn on paper. The exception is that the Game Box will color the grids for both current positions of each direction. Figure 1 is an example of the Game Box. In

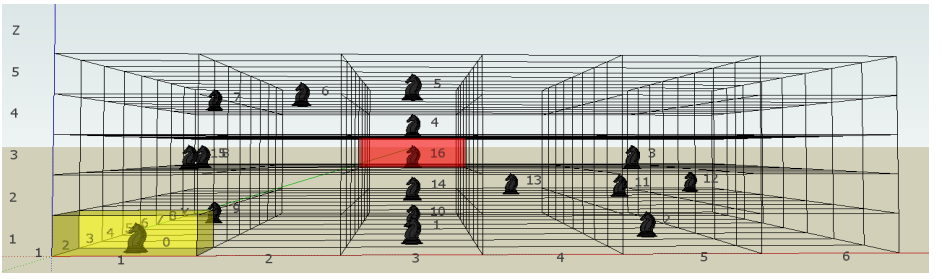

Fig. 1. Game Box View

this figure there is one grid colored red and another grid colored yellow. This is used to represent the two ends of the chain, corresponding to the two colored move tables (described below). By coloring the end points, it gives the user a clear indication of where the two ends need to meet in order to complete the game and which grids to avoid until every other position has been visited first.

2.3 The Window of 5x5x5

If the Game Box of l x m x n becomes very large, it poses difficulties for visualization and positioning in 3D, as the shading, coloring and grids can obscure and confuse the positions. A major feature that is designed to help the user to play the game is the 5^3 (or 5x5x5) window of the Game Box. This 5^3 window contains all the possible next moves for the knight's current (end) position. It is a tool for the user to make a more informed decision and can be made visible or invisible, i.e., turned on or off. Since its relation to the Game Box is always "rooted" at the knight's current position, it provides both a local view and a global perspective for the player.

There are two 5^3 windows, one for each end of the chain, distinguished by color. They can be displayed at the same time (possibly overlapping), or independently. At the center will be the current position, shown in Figure 2, which is marked by a red grid with a black knight piece and the number 16 next to it. Because of the unique movement of the knight piece there are only 24 possible legal positions that it can make from the center position. The positions that the knight is able to move to are shown in red. This is because the 5^3 window being shown is for the forward direction

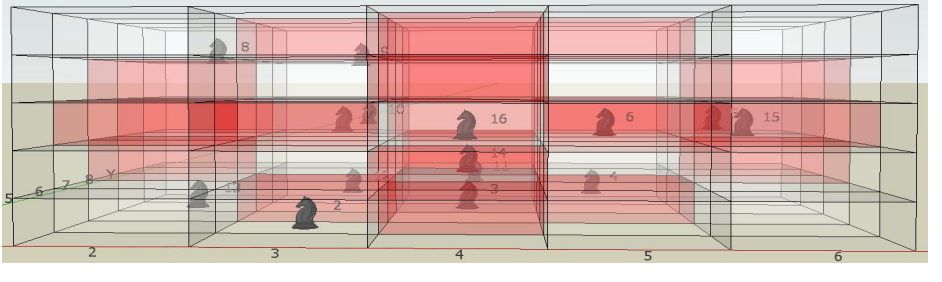

Fig. 2. 5^3 Window View

which is color coded as red. The backwards direction 5^3 window looks and functions in exactly same manner except that the current position is yellow and the grids that can be moved to in one knight move are also colored yellow. Positions that can be moved to but have already been occupied are colored gray and also contain a knight in the center.

The 5^3 windows will operate inside the Game Box as shown in Figure 7. As the next position is selected the current 5^3 window will center on the new position and all 125 grids of the 5^3 window will be updated with the new positions information. In Figure 7 there are different colored knight pieces, which indicate "move directions": a black knight is for the "forward" direction and a white knight is for the "backward" direction.

There are also special cases to the 5^3 window such was when the center grid is within two grids of the Game Box edge. This will result in some part of the 5^3 window being outside the actual game area. To resolve this, the illegal sections of the 5^3 window will not be visible outside of the game area and those outside points will not be selectable nor moved to. Another feature of the 5^3 window is that it displays those grids that can be moved to from either end of the move chain. Such grids are colored pink as seen in the Figure 3.

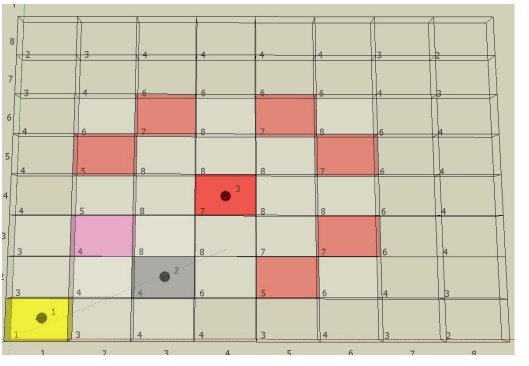

Fig. 3. Position Overlap

2.4 Selecting Next Legal Move

The LKT game environment allows the user multiple interactive options for selecting the next move. It also verifies that the next move selected is indeed legal, or else the user will be alerted and then given the opportunity to try another option.

With large Game Boxes the decision of picking the next move can be daunting and challenging with so many possibilities to track and visualize. The LKT game environment also keeps track of some useful counting heuristics (described below) to facilitate the decision (move) making process.

2.4.1 Text Input

Since the Game Box is a three dimensional grid system, each grid has an x, y, and z coordinate component. In order to select the next move the user has to input the x, y, and z value into the text boxes provided in the move tables. The use of the text input is more advantageous with larger 3D Game Boxes. This is due to the fact that the mouse's input, which will be discussed next, is only able to accurately select grids that are on the surface of the Game Box or relying on the user to zoom in to the critical 5^3 window. Therefore, if the user wishes to move to a grid that is not on the

outer surface, the user will have to "peel" away the grids that are in the way, obscuring a clear view. Not having to deal with this problem and just inputting the coordinates is a useful alternative.

2.4.2 Mouse Input

The mouse is used in the LKT environment for both overall controls and also for the selection of the next grid to move to. By clicking on the grid that the user wants to move to (when other grids are not in the way, or can be peeled away), the selected grid will be highlighted by a glowing blue outline. There is a confirmation button that must be pressed in order to actually move to that selected position. This insures that the user is confident with that move and that the mouse click was not a mistake.

A limitation of the mouse input is that it can only select grids that are on the surface of the Game Box or any 5^3 window. In order to select the inner grids, the view of just the 5^3 window can be used. Another method is to use the Section Plane tool [7] to peel away unwanted layers of grids until the desired grid is visible on the top viewing layer. With the unwanted grids removed for the time being, it is easy to select that particular grid by a mouse click.

2.4.3 Counting Heuristics

In order to successfully solve the LKT game, there needs to be some type of overall strategy that the user can heuristically rely on to make the next move and complete the loop solution. A few simple counting of open neighbors are provided in the visual display to track the current state and look ahead a step or two to help the user continue the play. It is still the user's decision to follow them or how to use them.

The Count of Open Successors (COS) is the number of open grids that the knight in the current position can move to. The grid with the lowest COS can be chosen as the next move. This criterion is derived from the most constrained (MC) strategy of [10, 11].

For such counting, it is inevitable and often for many grids to have the same integer COS. For grids tied with the same lowest COS, another counting estimate of Neighborhood Flexibility Count (NFC) is computed and displayed. The NFC value for each tied least-COS grid is the sum of all the COS values for each of its neighbors. Then the grid (among those tied) with the largest NFC value can be chosen as the next move. This criterion is derived from the least impact (LI) strategy of [10, 11].

Even after applying the tie-breaking NFC, it is still possible to have some grids tied with the same COS and NFC values. The tie breaking method is the Loop Targeting Measure (LTM). First, the current chain length is subtracted away from the ultimate loop length *lmn* to compute how far away from the current chain is to the final loop solution. The second estimate is the number of possible (open) grids that exist between the other end of the current chain and the tied grid. The LTM value is then computed as the difference between the first value and the second estimate. Such an LTM value is designed to provide an estimated measure to the goal of the final LKT solution loop, i.e., a heuristic estimate of the "distance to the goal state". To break the NFC tie, the smallest LTM value among the tied grids can be chosen.

If there is still a tie after applying these counting and tie-breaking values, then the final pick is a random grid among the remaining tied grids. These counting heuristic are merely computed for the convenience of the user and serves only as guidelines

that can be deviated from at any step. They can and have helped to achieve some LKT solutions in one straight sequence of move selections without any backtracking. But they can, also, lead to dead ends that will require backtracking or look-ahead to complete the solution loop. But, it is clear that no heuristic can be infallible, so that these three values are computed and tabulated in the table of Neighbor Pool on the LKT game display, an example of which can be seen in Figure 5.

2.5 Move Tables

Solving the LKT problem is a process of extending a chain of moves (until the final closed loop). There are always two directions to extend the ends of the chain. Thus, there are two move tables, one for each of the directions. The starting position is move number 1 in both tables, from which the ends of the chain are extended as the tables grow. The Red move table is designated "forward" and the Yellow table is considered in the "backward" direction. The move tables contain the past movements sequentially in their respective directions, and can be scrolled down to check for any past positions. The more recent movements will put on the top of all previous moves. Figure 4 shows both move tables; however in the LKT game environment they are on opposite sides of the display. Blank boxes in each table allow the user to input the coordinates of the next move. There is also the "CheckLoop" button to test if the sequence of moves so far (in one of the two tables) is a loop or not. The "CheckLoop" button is not so useful until the two sequences are combined, since the move number 1 is the starting point for both sequences and only the last position in each table could possibly form a loop. Its value will be shown when the sequences in the two tables are combined by using the Control Panel (below) and the possibility of achieving a loop can be checked in one Move Table.

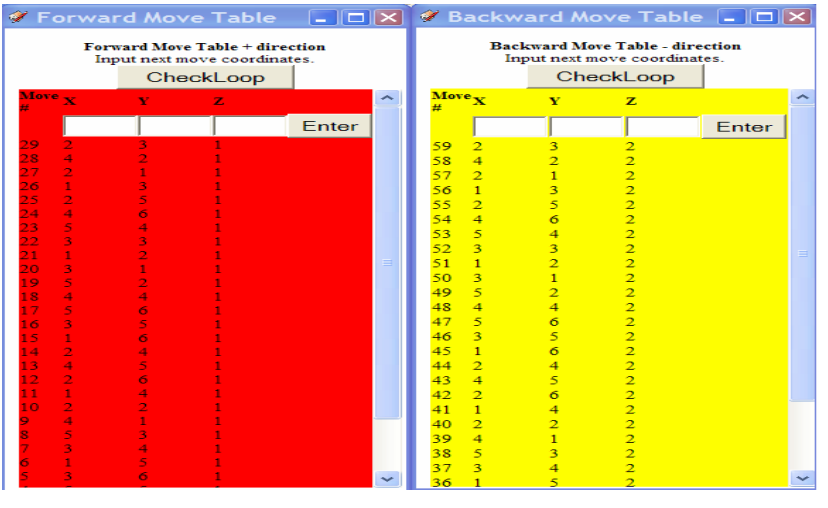

Fig. 4. Move Tables

2.6 Neighbor Pool

The neighbor pool contains a list of next move coordinates for each of the ends of the chain. It also contains the heuristic values of the COS, NFC and LTM for each grid with (x, y, z) coordinates. Figure 5 shows that the lines (coordinates) are colored according to which Move Table (or direction) they came from. The pink grids correspond to positions that can be reached by both ends of the chain. Each of the column labels of the Neighbor Pool can be clicked on, to sort the values in that column in increasing or decreasing order. For example, the user can sort all the next move positions by the COS value, to make it easier to select the move with the least COS. Then, sorting by the NFC and LTM columns simplifies the tie-breaking processes. One feature to easily improve the quality of game play is that of pointing and selecting a row, hence the coordinates as the next move.

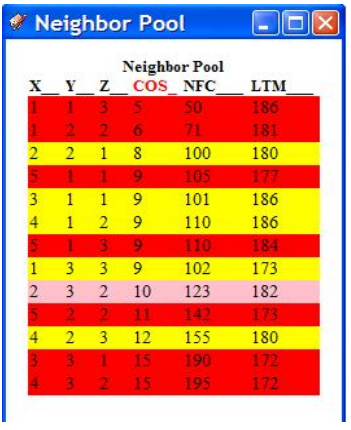

Fig. 5. Neighbor Pool

2.7 Control Panel

The Control Panel (Figure 6) is where the user makes most of the interactions with the game. At the top of the control panel is the current direction information, shown either in red or yellow to keep track of which end of the chain is currently active.

The user is able to save the current game in progress to a file by pressing the "Save" button. The saved chain or loop can be loaded by the "Load" button. The user can also load another file to perhaps combine two solutions together in order to get a larger solution by pressing the "Load" button.

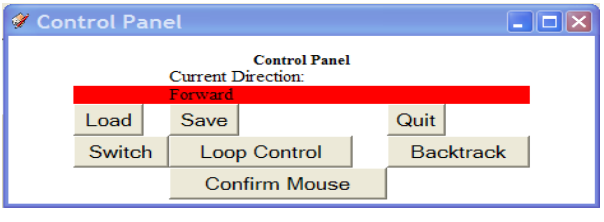

Fig. 6. Control Panel

The "Backtrack" button allows the user to move back from the current position to the previous position, or to repeatedly backtrack several steps. Any backtracked move is forgotten, i.e., no history is kept so as to allow stepping forward again.

Another button in the control panel is the "Switch" button to change the direction (end) the moves are being made. Clicking the "Switch" button toggles between the forward (red) and backward (yellow) tables as the active direction. The current direction text and color will change accordingly to show the current direction or table.

By pressing the "Loop Control" button the user is then presented with the Loop Control window which has the ability to break a loop, reorder a loop, or combine two loops together to get a larger solution which is called stitching in 2D and knight's crossing in 3D. In the Loop Control window the user has five text boxes to fill in. The first and third text boxes are for the direction of loop that is being worked on. The second and fourth text boxes are the move numbers, and the last text box is to reverse the counting of the loop. The Loop Control breaks the loop at the designated point such that the first move and final move is where the break point occurs. This creates a chain of movements that can be combined with another chain of movements to create a larger chain of movements. After the two chains have been linked together into a larger chain, the "CheckLoop" button is then used to verify that the combination of the two chains does result in a new larger loop solution.

The "Confirm Mouse" button is used with Mouse Input (described above) to reaffirm that the mouse-selected grid is indeed the move that the user wants to make.

3 LKT Game Environment

Playing a 2D LKT game might be relatively simple (and theoretically solvable for any rectangular board [6]), since the entire board is viewable as seen in Figure 3. In a 3D Game Box, however, more viewing perspectives and manipulating tools are required and counting information (such as those provided in Sec. 2.4.3) to assist the user to monitor the current situation becomes more essential. The combination of the table displays with position information and the visual manipulation tools facilitates tracking during play. The ability to track and view positional relations, ranging from the local grid associations in the 5^3 windows, to the global perspectives offered by the entire Game Box, are what ultimately enables the user to understand the present situation and make decisions for the next move. Such capabilities, including the Neighbor Pool table that track the counting heuristics also facilitate user's understanding of the global relations (between the two ends of the chain in the "end game" and offer feasibility for completing a LKT loop.

In order to model, view and appreciate a 3D body with interior details through a 2D medium there needs to be a way to view the 3D object from all angles. Figure 7 is a sequence of snapshots taken as the Game Box is rotated in the counterclockwise direction. It illustrates a middle game stage where the user has made a multitude of moves in each of the forward and backward directions (tables). Both 5^3 windows are visible, and distinguished by the red and yellow colored grids. The difference between the sub-figures is a change of the orientation of the Game Box in the x-y plane, while the z plane remains constant. The axis labels also change as the view is changed to always be in the correct orientation to be read properly.

Being able to change the view of the Game Box is important in order to accurately assess the overall situation and make the best possible choice for the next move. It also allows the planning and strategizing of moves before committing to one. For example, in order to select a grid on the interior of the Game Box the user can to either input the coordinates into the appropriate "Move Table" or use SketchUp's built in "Section Plane" tool to peel away the grids that inhibit the mouse from

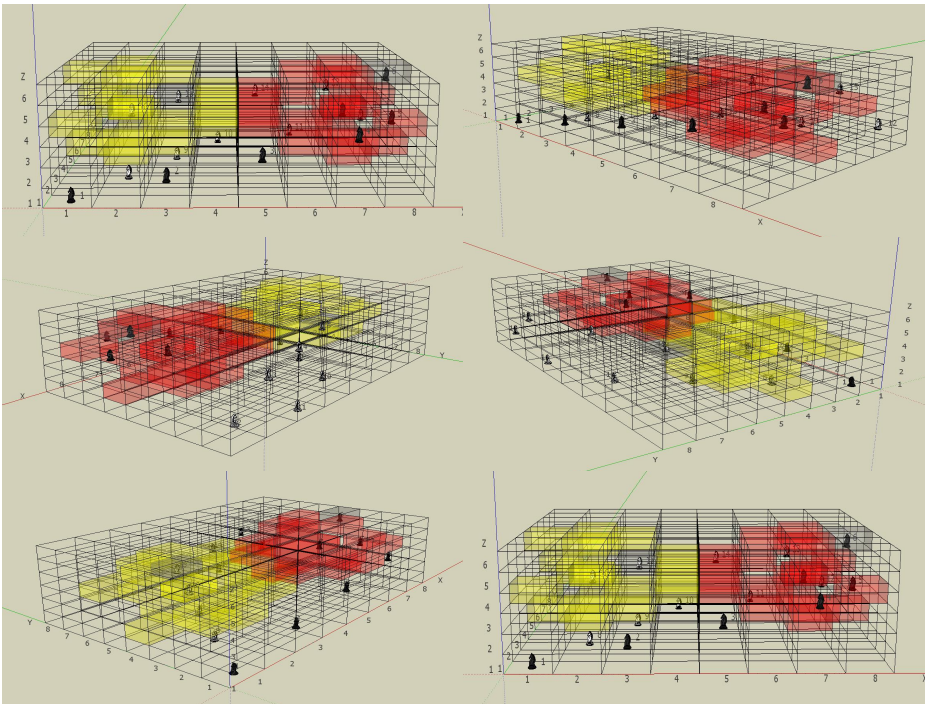

Fig. 7. Snapshot sequence showing the rotation of the Game Box in 3D (left to right, top down)

selecting the grid. Even by using the "Section Plane" or the 5^3 window, the viewing perspective may need to be changed in order to be able to see and select the appropriate grid.

4 Conclusion

Having a visual representation of problem states has demonstrated its power of a clarified view and understanding so as to enable the user to cut through the complexity of these problems and decide on the most logical steps to take. Furthermore, a visual system can stimulate different methods of thinking and attacking problems. This will, in turn, prompt new ways of approaching the same problem with more effectiveness or efficiency.

4.1 Solving the LKT Problems

The graphic environment for LKT game playing in 3D is supported and facilitated by the ***LKT game environment*** described above. The LKT game operates within Google's SketchUp 3D modeling software and utilizes Ruby, Java and HTML programming languages [7] for several features and extensions. Trying to play a LKT game mentally, without visual aid or computer assisted tracking, is extremely

difficult, if not impossible, because the user must keep track of all position and move possibilities from both local and global perspectives in 3D. Using the LKT game environment, many variables (and open and occupied positions, and their relations) can be monitored and displayed for the user. The value of such a 3D graphic environment and computer assistance is, thus, demonstrated.

The overall solution approach towards a complete collection of solutions for all the 3D LKT problems will require a community of players challenging one another and cooperating to develop an increasing database of solutions. As smaller solutions are found and stored in a Web searchable database, larger sizes posed online can be solved by partitioning the size into smaller 2D layers or 3D blocks, some of which may have existing solutions, and composed together by 2D stitching or knight's crossing in 3D. Such a combination of top-down partitioning and bottom-up solution database yields a much more realistic, and interesting, solution method to solve all LKT problems in 3D, rectangular boxes or other shapes. Any time a larger size problem is solved; the database of solutions would not only index and store its loop solution, as a chain of knight's moves, but also its specific decomposition into smaller sizes. Such an indexed database developed by a community of players on the Web would grow with each new solution found and new game posed. The Web player community can blossom by using this LKT game environment via SketchUp and Google's world-wide distribution to achieve complete set of LKT solutions, while playing and enjoying the challenging game online.

4.2 Assessment of Potentials

3D games visually aided by motion, zooming and slicing and effectively assisted by computed tracking and counting information, facilitate the user decision making without diminishing the pleasure or challenge. The drawbacks of having to deal with 3D scenes and objects on 2D screens can largely be overcome. Computed assistance and visualizations can also help overcome certain physical difficulties to make the process of problem solving more interesting and enjoyable. The LKT game is such a demonstration of a computer enhanced environment to enable enjoyable playing of a 3D game.

The LKT environment can also be utilized to develop other types of games. There are games that claim to offer 3D chess, but in reality only the chess pieces are 3D while the game is still shown on a stack of 2D boards [1]. Due to the lack of computed tracking values and counting displays to facilitate decisions, playing such a 3D chess game is not much more than a stretch of the mental capacity beyond the 2D game. The LKT game environment described/offered here can be easily modified to support 3D chess and possibly trigger renewed interest worldwide via the Web.

Many games, including chess, are already played online. The 3D model system can offer a multitude of computed tracking to enhance the enjoyment, but maintain the usual challenges. The LKT game cannot be put into a physical realized model hence is only appropriate as a computerized game. As such, it has demonstrated the necessary 3D feasibility and added enjoyment and challenge through computer assistance and displays.

References

1. Bornet, R.E.: The Game of Three Dimensional Eight Level Chess (2004), http://www.hixoxih.com/games/chess/3D8L.htm#TRUE%203D%20CHESS
2. Duffy, S.: How to Do Everything with JavaScript. McGraw-Hill, Berkeley (2003)
3. Fitzgerald, M.: A Quick Guide to Ruby Pocket Reference. O'Reilly Media, Inc., Sebastopol (2007)
4. Flanagan, D., Matsumoto, Y.: The Ruby Programming Language. O'Reilly Media, Inc., Sebastopol (2008)
5. Koosis, D., Koosis, D.J.: Java Programming for Dummies, 2nd edn. IDG Books Worldwide, Inc., Foster City (1997)
6. Schwenk, A.J.: Which Rectangular Chessboards Have a Knight's Tour? Mathematics Magazine, pp. 325–332 (December 1991)
7. SketchUp. Ver. 6. Google (2008), http://sketchup.google.com/
8. Stewart, I.: Math Recreations Knight's Tours. Scientific American 4(97), 102–103
9. Thomas, D., Fowler, C., Hunt, A.: Programming Ruby The Pragmatic Programmers' Guide, 2nd edn. The Pragmatic Programmers, USA (2005)
10. Yun, D.Y.Y.: Achieving Computational Intelligence by Resource Optimization (Invited Paper). In: Proc. 2002 World Congress on Computational Intelligence. International Joint Conference on Neural Networks, Honolulu (May 2002)
11. Yun, D.Y.Y.: Intelligent Resource Management through the Constrained Resource Planning Engine. In: Kasabox, N. (ed.) Future Directions for Intelligent Systems and Information Sciences, ch. 18, pp. 373–386. Springer, Heidelberg (2000)