Design and implementation of Surakarta game based on iOS

Yizhong Qi\textsuperscript{1}, Shuqin Li\textsuperscript{1,3,*}, Jianbo Bo\textsuperscript{1}, Yao Fu\textsuperscript{1}

\textsuperscript{1}Computer Academy, Beijing Informations Science & Technology University, Beijing, China
\textsuperscript{2}Perception And Computation Intelligence Joint Lab, No.35, north fourth ring road, Chaoyang district, Beijing, China

*Corresponding author e-mail: lishuqin_de@126.com

Abstract. Aiming at the present situation that Surakarta game has a narrow audience and poor user interface and user experience, this paper designs and implements a Surakarta game system based on iOS platform. A chessboard data structure based on iOS platform is designed, which is suitable for different screen sizes of mobile phones. The animation representation of game process, especially the eating process, is realized. This paper studies and attempts to use GCD-based multi-threaded search to solve the problem of limited mobile phone resource performance. Experiments show that the Surakarta game system based on iOS platform can play man-machine game on portable devices and has good performance.

1. Introduction

Surakarta game is a two-player game and is one of the events of ICGA [1] and Computer game competition for Chinese University Students [2]. Its chessboard is made up of 36 points and 8 circular arcs, as shown in Figure 1. The two sides of the game are divided into the upper and lower two parties' camps. Each side has 12 pieces, each of which moves one square of eight moves on its way up, down, left, right, left up, right up, left down and right down. In walking, if there are already chess pieces of one's own side or the other's enemy, the landing is invalid. In the course of flight, the Eater must circle at least one circle and not be blocked by his own chess pieces in the arc. The condition for judging victory is a chess piece that one side eats the other side. Because there is still a situation in the game of chess cannot finish the opponent's chess, so in this case the side with more chess won. At present, Surakarta game only appears in the intelligent game competition in China, and operates on the PC side, so the audience is very small, and the user interface and user experience effect are somewhat different from the current mainstream form of interaction, Therefore, Surakarta game is hardly known.

iOS system is one of the most popular mobile operating systems in the world. The design and implementation of Surakarta game on iOS system can effectively improve the audience of Surakarta game players and get rid of the single PC-side machine game. However, the difficulty of implementing Surakarta game on mobile phone is that the chip performance and memory space of mobile phone are limited. In iOS system, an application that occupies too much memory will be killed by the system. The purpose of this paper is to research and implement Surakarta game based on iOS platform, improve the level of user interface and user experience, and use limited mobile phone performance to improve the design of Surakarta game AI level.
2. Design of Surakarta game based on iOS

2.1. Storage structure of Surakarta game
Designing Surakarta game first needs to establish a good data structure for storing chessboard. In design, a chessboard is represented as a matrix of 6x6. As shown in Figure 1. In order to achieve the two effects of Surakarta game moving and flying, each position object YZChessPlace in the matrix has six attributes: tag, camp, x, y, frame X and frame Y. As follows:
@interface YZChessPlace : NSObject
@property(assign,nonatomic)NSInteger camp;
@property(assign,nonatomic)NSInteger tag;
@property(assign,nonatomic)NSInteger x;
@property(assign,nonatomic)NSInteger y;
@property(assign,nonatomic)CGFloat frameX;
@property(assign,minimal)CGFloat frameY;
@end

1) The tag attribute is the unique annotation of each chessman from 1 to 24. 1 to 12 tagging the upper camp, 13 to 24 marking the lower camp pieces.
2) The camp attribute is used to mark the position of the camp. If the camp attribute value of the upper camp is -1, the camp attribute value of the lower camp is 1, and if there is no chessboard in the position, the camp attribute value is 0.
3) The X and Y attributes are used to represent the position values of chessmen in the chessboard matrix. X represents the longitudinal coordinates of the chessboard, and Y represents the abscissa of the chessboard, with a range of 0 to 5.
4) The frameX and frameY attributes are used to represent the abscissa and ordinate coordinates of a chessboard in the UI. The range of values is determined by the screen size of the mobile phone. They are used in walking and flying methods.

2.2. Chessboard UI design and Implementation
The checkerboard UI is designed using the Bessel curve [4]. The chessboard UI can be split into a wireframe, 4 vertical lines, 4 horizontal lines and 8 3/4 arcs. By rewriting the chessboard View, the following methods are implemented:

- (void)drawRect:(CGRect)rect;

In the drawRect method, the center of the screen is taken as the benchmark point, and moveToPoint method is used to move to the upper left corner of the chessboard according to the ratio of screen widths. Then the addLineToPoint method is used to draw the outermost wireframe. Back to the origin of the upper left corner, move 30 screen points to the right each time, and draw a vertical line using the moveToPoint and addLineToPoint methods. The horizontal line is the same from top to bottom. The 8 curves are 3/4 arcs, which need to be moved to the 4 vertices that the chessboard has drawn. The bezierPathWithArcCenter method is used to draw 3/4 arcs with the top angle as the center, 30 or 60 as the radius, so that the vertical and horizontal lines of the track are connected with the inner and outer rings.

Considering that the UI needs to be adapted to different phone models, the length and size of the board UI should be calculated in proportion to the screen. The method used in this paper is to start from the center of the screen, each grid composed of vertical and horizontal lines occupies 30 screen points. Such a design can occupy a more reasonable size in the screen and is suitable for all kinds of mobile phones. The result is shown in Figure 2.
2.3. Design and implementation of eating algorithm

The algorithm adopts iteration and recursion. When the chessman receives a click event, it iterates in four directions: the upper, the lower, the left and the right.

For example, if you click on a piece in the (4,1) upward direction, first move up a point to determine whether there is a piece, if there is a piece, whether we or the enemy, it means that there is no upward direction of flight law. If there is no chessman, continue to look up until the critical point. Key points refer to the position of each arc in and out of the arc, which correspond to (0,1), (0,2), (0,3), (0,4), (1,0), (2,0), (2,5), (3,0), (3,5), (4,0), (4,5), (5,1), (5,2), (5,3), (5,4) of Figure 1, respectively. Here is (0,1). If there is no chessboard obstruction in the upward traversal, the key points (0,1) of the arc into the arc and the key points (1,0) of the arc out are added to the pre-set flight array. Here, this piece of chess has formed the condition of flying to eat by ascending and then flying around. Then traverse to the right from the key point (1,0) of the arc's exit. If you first encounter our chessboard, the flying method is blocked. Exit the upward traverse and choose an untraversed direction to continue. If the enemy is chessmen, the position of the enemy pieces is added to the flight array to exit the upward direction. If it has not been blocked, continue to repeat the steps through the arc.

After all traversal, considering that it is possible for a piece of chess to fly and eat more than one piece, the final outgoing structure should be a two-dimensional array in which the one-dimensional array contains each eatable path, where the path is made up of key points, which are mapped to the code as the data class mentioned above.

The data structure is handed over to the animation method, and the animation method splices the Bessel curve through the key points to complete the process from eating to flying eating. The computation efficiency of the algorithm is only 0.0001s in real iPhone 8 test. The specific steps are as follows:

**Step 1:** Received click events, if the pieces in the corner vertices (0,0), (0,5), (5,0) and (5,5) then exit, there is no case to eat.
**Step 2:** Select the four directions of the upper and lower directions to traverse without traversal. No direction can be traversed and exit.
**Step 3:** There is a chess piece in this direction, back to step 2. If there is no, the key points and key points of the arc are added to the flight array.
**Step 4:** The exchange direction continues to traverse.
**Step 5:** In the current direction, there is our chessboard back to step 2, and when there is an enemy, the position of the enemy's chessboard is added to the flight array.
**Step 6:** Add the flight array to the two-dimensional array of edible seeds and go back to step 2.

2.4. Design and implementation of flying animation

According to the rules of Surakarta game, if you eat the other side, you must first walk around the lake for at least one lap. In this paper, we use animated animation to call the CAKeyframeAnimation class.

+ (CAKeyframeAnimation *)animationWithChessCenter:
(CGPoint)center chessArray:(NSArray*)array;

The first parameter in the function is the center position on the UI before the board flies, and the second parameter is an array of routes passed in. The array of flight paths is made up of key points. The vertical, horizontal, and curved Bessel curves can be joined together by the 16 key points of the incoming flight array parameters which constitute the necessary key points for flight. In each stitching, you also need to accumulate the appropriate animation time and use the appropriate timingFunction. To achieve the following, the variable path is the Bessel curve that is spliced by the incoming. Through the frame animation, the pieces move along the Bessel curve of the chessboard. At the beginning of the animation, the layer of the pawn is moved first, and the frame value of the chessman is changed in the callback of the animation. The animation effect is shown in Figure 3, as shown in Figure 4. Figure 3 shows the blue side attacking the inner ring of the red side. Figure 4 shows the red side's son 21 flying to eat the blue side's son 8 in 48 seconds.

CAKeyframeAnimation *animation = [CAKeyframeAnimation animation];
animation.timingFunction = [CAMediaTimingFunction functionWithName:kCAMediaTimingFunctionLinear];
animation.keyPath = @"position";
animation.duration = duration;
animation.path = path.CGPath;
animation.removedOnCompletion = NO;
animation.fillMode = kCAFillModeForwards;

3. Design and implementation of Surakarta AI algorithm
In mobile devices, the performance of the chip is far less than the performance of the PC chip, the program occupies more than 1G memory will be killed by the system, so the design of AI needs to consider the power of chess and algorithm efficiency, optimize memory usage, timely release of memory. At present, Surakarta game uses alpha-beta pruning search algorithm in most of the game. The AI search algorithm in this paper refers to the alpha-beta pruning algorithm used in reference [3], and on the basis of which the relevant optimization for mobile phone performance conditions is completed.

3.1. Opening strategy design
The study found that the opening of Surakarta game is very important for the result of the whole game. Therefore, in the beginning, we should pay more attention to strategy rather than search. On the first hand, we can go ahead and give priority to the inner attack. The red side is shown in Figure 3. The red side (1,5) moves towards (2,5), (14) towards (2,3), (0,4) towards (1,4) to initiate the first-hand inner ring attack. The top-corner pieces should be removed in time after attacking, so as to avoid neglecting the top-corner pieces in the process of changing the pieces, which makes the top-corner pieces become dead in the middle period, that is, unable to move, moving will be eaten passively.

At the beginning of the backhand, you need to guard against the enemy's inner ring attack, which is a very common start, as shown in Figure 3. In this case, the lower right-hand corner into the arc or out of the arc (4,5) and (3,5), so that the defensive position from the inner ring extended to the inner ring and outer ring together defense. In this case, if the enemy still insists on replacing the inner ring, the inner ring will be controlled by us. In Surakarta game, once the inner ring is controlled, two pieces in the top corner of the opponent become dead, and the opponent's chess power drops sharply. If the enemy does not choose to attack at this time, you can also draw with the enemy in the defensive side of the first-hand advantage, rather than the final change is always less than the enemy a chess piece. If the enemy is unwilling to change his son after the change, we will not tie down the first-hand advantage, but a successful defense, at this time it is necessary to open up the battlefield on the left as soon as possible. In the beginning of the backhand, there may also be a situation in which the other side's intention is not clear. At this time, we can choose to oppose the guest mainly and launch the outer ring attack.
3.2. Design and implementation of multi thread search and search algorithm

The search uses alpha-beta pruning, depth 4 to 6 layers, and evaluation function reference [3]. AI's search at the first level needs to be tailored directly to the situation without searching. There are 2 situations:

1) The situation will be eaten directly. In Surakarta game, the first to be eaten is in a disadvantageous situation, excluding the excessive advantage of one's own side deliberately seduced to eat, the situation of being eaten directly and subsequent operations are not necessary.

2) At the beginning of the game, we entered the top corner. It is wrong to go into the top corner at the beginning of the game from the first step, and the situation of subsequent search based on this step need not be considered. So, we need to cut it down on the first floor.

The AI algorithm in this paper pays more attention to eating and swapping, so in the first level of search, if you can eat the situation will be given a reward score, this reward score must be given very carefully, because in many cases eating is not necessarily the best choice. The AI in this paper has given the reward points still do not give priority to eating, which is the role of the evaluation function. If the AI occupies the Inner or Outer Ring intersections in the first step, reward points are also required to enable the AI to attack again.

In order to achieve better search efficiency, this paper uses GCD technology to search [5] by multithreading. After the initial trimming of the first level, enter the for cycle. For each loop, a thread is opened for parallel operation. In the process of operation, the core data exchange needs to be locked, in this performance-sensitive algorithm, the use of the lock also needs to consider the efficiency. The dispatch_semaphore_t is used in this paper. The semaphore is a reliable and excellent lock in iOS development. Because alpha-beta searches have such an operation as impersonating and withdrawing, it is necessary to lock in the process, otherwise the chessboard will be confused, and subsequent calculations will be wrong.

4. AI call interface design

In engineering design, it is necessary to keep code low coupling and high cohesion. The design of the AI interface needs to follow a line of principle, that is, one line of code can call the AI algorithm. Considering that the game AI algorithm needs to change the parameters according to the chessboard in the middle of the game, this paper modifies the AI parameters by changing the Controller attribute. The following are the AI interfaces and properties:

- (NSDictionary *)stepDataWithChessPlace:(NSArray *)chessPlace;
- @property(nonatomic, assign)NSInteger camp;
- @property(nonatomic, assign)NSInteger searchDepth;
- @property(nonatomic, assign)NSInteger stepNum;
- @property(nonatomic, assign)BOOL isFirst;
The camp attribute indicates that AI is the upper or lower camp; the searchDepth attribute is the search depth, and AI needs to modify the search depth dynamically according to the chessboard; the stepNum is the next step in recording AI; the isFirst is the first step in recording AI, and the evaluation function needs to be adjusted according to the first step.

The AI algorithm is invoked through the stepDataWithChessPlace method into the checkerboard. This method returns a dictionary type that contains AI's method type, evaluation value, and the pieces AI needs to play. The calling method can also be designed as a form of block callback, as shown below:

- (void)stepWithChessPlace:(NSArray *)chessPlace
  block:(void(^)(NSDictionary *step))block;

When using the above interface, you need to create a separate thread for AI and return to the main thread to complete the UI operation after completing the calculation.

5. Experimental results and analysis

This program is designed and implemented in Objective-C language under the environment of the iPhone 8 simulator of MacOS system Xcode 9. In this paper, we do two experiments, one is to test the function of iOS-based Surakarta game, the other is to test its performance.

Experiment 1: the function of Surakarta game based on iOS

Based on iOS's Surakarta game system, this paper implements two functions of man to man and man to machine. After choosing the "start" game, the player can choose everyone to fight and man-machine confrontation. As shown in Figures 4 and 5. In the "Settings" feature, players can choose AI on the red or blue side and turn on sound, vibration and special effects. As shown in Figure 6. Figure 7 and figure 8 test the process of attacking the red square pieces by the blue side chessmen. Figure 7 shows the blue side attacking the inner ring of the red side in 07 seconds, and Figure 8 shows the blue side 20 flying to eat the red side 2 in 8 seconds. The test shows that the algorithm is stable and fast, and the flight animation is running normally. AI can conduct normal search and evaluation on red or blue side.
Experiment 2: play chess with the sula2016 program running on the PC side of the same school.

This procedure and sula2016 play chess in 10 games, lose 5 games, tie 4 games, win 1 game. The result of playing chess is shown in Table 1. The average time of the game is 8 minutes, which meets the requirement of 15 minutes or less in Computer game competition for Chinese University Students. In this program thinking about playing chess, mobile phone CPU occupancy remains between 90% and 100%, using the full performance of the mobile phone chip. The use of memory is stable within 120M. There are no serious problems such as collapse during long time operation. This paper achieves the Surakarta game in the Chinese University Game Competition Surakarta project has won the third prize results, verifying the feasibility of the program.

| Game | Game 2 | Game 3 | Game 4 | Game 5 |
|------|--------|--------|--------|--------|
| Our Start at first | Lose 3 | Lose 3 | Draw   | Win 1  |
| The other side first | Draw   | Draw   | Draw   | Lose 1 |
|                  |        |        |        | Lose 1 |

6. Conclusion

Based on the iOS platform, this paper focuses on user interface and user operation, and implements a computer game system on portable devices, which can play the game between human and machine, and has good performance. Limited to the performance of mobile phones, in the same algorithm and pruning strategy, the PC-side Surakarta AI search depth is far beyond the mobile end, mobile phone Surakarta AI to compare with the PC-side Surakarta AI chess strength, but also need further improvement. The author plans to further study in the following aspects: optimizing the evaluation function and improving the search depth.

Acknowledgments

This work is supported by 2018 Talent-Development Quality Enhancement Project of BISTU (5111823402), supported by the special bidding project of teaching & education reform (2017JGZB08), and Supported by National Natural Science Foundation of China (No.61502039).

References

[1] International Computer Games Association on https://icga.org
[2] 2018 Computer game competition for Chinese University Students on https://www.saikr.com/caaigames/bistu/2018
[3] ShuQin Li, JingBo Li, YuHua Han, The assessment function in the Surakarta game system[J], Journal of Beijing Information Science and Technology University (NATURAL SCIENCE EDITION), 2012,27(06):42-45+61.
[4] YaLi Luo, Custom control -- Bessel curve[J], Information and computers (theoretical version), 2010(06):109-110.
[5] Wei Chen, QinKai Bo, Research on Multithreading Technology in iOS system[J]. Computer knowledge and technology, 2017,13(08):78-80.