2014 IEEE Sixth International Conference on Technology for Education

CarromTutor: A cognitive apprenticeship based tutor for carrom skills and strategies

Mrinal Malick
Dept. of Computer Science and Engg.
Indian Institute of Technology Bombay
Mumbai, India 400076
Email: mrinal1090@gmail.com

Mayur Katke
Dept. of Computer Science and Engg.
Indian Institute of Technology Bombay
Mumbai, India 400076
Email: katkemayur2@gmail.com

Sridhar Iyer
Dept. of Computer Science and Engg.
Indian Institute of Technology Bombay
Mumbai, India 400076
Email: sri@iitb.ac.in

***Abstract*—Carrom is a popular game and there are many online and standalone applications for playing it. However there is no tutor for systematic teaching-learning of Carrom skills and strategies. In this paper, we describe the design, implementation and evaluation of *CarromTutor*, for teaching-learning of Carrom skills and strategies. *CarromTutor* incorporates principles of cognitive apprenticeship and game-based learning, and is implemented in Blender, a free and open source 3D animation suite. We evaluated *CarromTutor* through a study involving both novices and experts. We measured learning using pre-post exercises, and usability by administering the SUS scale. We also compared *CarromTutor* with *Carrom King*, a popular online version of the game. We found that (i) most learners were able to learn 3-6 new skills and found it easy to relate the tutorials to the exercises, (ii) the SUS rating was 84.09 (grade A), and (iii) learners gave positive feedback on the interactivity, 3D point of view, and teaching-learning approach in *CarromTutor*.**

I. Introduction

Carrom is a "strike and pocket" table game of Eastern origin similar to billiards [1]. The game is usually played on a board made of plywood, by two players (singles) or two teams (doubles). The objective is to flick a striker disk towards lighter disks and land them into one of four corner pockets. Many online Carrom games are available which simulate a carrom board and allow players to use a keyboard, mouse, or touch, to play the game. For example, *Carrom King* is played using the mouse [2]. In this, the striker is placed by moving it to the desired location using the mouse and left click. Then dragging the mouse determines the direction and force of the flick.

Carrom requires skill as well as strategy. Carrom skills, ranging from straight shots to advanced deflection shots, and strategies for break and block, are discussed in [3]. A novice typically learns these skills and strategies implicitly by directly playing the game and by observing expert players. While this is a valid teaching-learning method, an approach that makes the cognitive strategies explicit has many benefits. Explicit instruction refers to an instructional practice that carefully constructs interactions between students and teacher, and has positive outcomes for students [4].

We have designed and implemented *CarromTutor*, an online game for explicit instruction of Carrom skills and strategies. A screenshot of a user playing an exercise is shown in figure 1. *CarromTutor* incorporates principles of cognitive apprenticeship [5], such as modeling, sequencing and scaffolding, and guidelines of game-based learning [6], such as immersive engagement, problem-based learning structures and feedback. This is explained in Section 3.

Fig. 1: User view of an exercise in CarromTutor

CarromTutor first provides a video tutorial of specific Carrom skills, such as, coin to coin deflection, double touch, etc. Then it provides practice exercises for the learner to play, which may be repeated till the learner attains mastery. Finally, it provides learners with complex exercises wherein they can apply multiple previously learned skills in a game-like environment. This is explained in Section 4. *CarromTutor* is implemented in Blender [7], a free and open source 3D animation suite. Blender supports modeling of 3D objects, animation, simulation, and rendering, for game creation. We have implemented *CarromTutor* using the Logic editor and by writing Python scripts in Blender as explained in Section 5.

We have done a preliminary evaluation of *CarromTutor* through a study having five beginners, five intermediate players and one expert. We measured perception of learning and perceived ease of pre-post exercises. Most participants reported learning of 3-6 new carrom skills in a first exposure to *CarromTutor*. They rated the exercises as 'Hard' before the training, and rated similar exercises as 'Easy' afterwards. We measured usability of *CarromTutor* using the SUS scale [8]. Analysis of the participants responses resulted in a SUS score of 84.09 (grade A); a score of 68 is considered average [9]. Participants also compared *CarromTutor* with *Carrom King*, and gave positive feedback on the interactivity, 3D point of view, and teaching-learning approach in *CarromTutor*.

978-1-4799-6489-5/14 $31.00 © 2014 IEEE
DOI 10.1109/T4E.2014.20

II. Related Work

There are many online Carrom games, such as Carrom King [2]. These focus on enabling the user to play the game, rather than teaching-learning of the skills and strategies. A novice typically learns carrom skills and strategies implicitly by directly playing the game. However, an approach that makes the cognitive strategies explicit has many benefits. Explicit instruction refers to an instructional practice that carefully constructs interactions between students and teacher, and has positive outcomes for students [4]. To the best of our knowledge, there is no online game (or tool) designed for explicit instruction of Carrom skills and strategies.

We looked at teaching-learning of other games. Chess teaching manual [10] explains each move to user with text explanation and diagrammatic representation. Initially basics of the game are explained, followed by details of strategy for playing a move in a specific situations. Then some situation is presented and the learner has to identify the appropriate move for that situation. Subsequently, explanation for the correct answers is given to learners in a step by step manner, considering may possibilities. The working of *CarromTutor* is along similar lines, while also incorporating tutoring and assessment.

We also considered a variety of teaching-learning methods and found Cognitive Apprenticeship [5], and Game-based Learning [11] to be suitable for the design of *CarromTutor*. These are explained in the next section.

III. CarromTutor: Design

In order to build *CarromTutor* we began with the following questions:

- **Q1:** Which Carrom skills and strategies are to be included?
- **Q2:** Which teaching-learning methods are to be used to teach these skills and strategies?
- **Q3:** Which features of game-based learning are to be used?

We answered question Q1 above by referring to literature on Carrom. A list of Carrom skills and their description is given in [3]. One of the developers of *CarromTutor*, who is also a National level Carrom player, classified these skills in increasing order of difficulty into:

- Basic skills: Straight shot, Normal cut, Straight cut, Negative cut, Doubling.
- Intermediate skills: Punch, Rebound, Press, Connection.
- Advanced skills: Coin to coin deflection, Striker's deflection, Cut return, Double touch, Follow, Coin on baseline, Rolling of striker.
- Strategies: Break, Queen, Negative game, Easy cover, Safe game, Check block, Game pass.

All of the above skills and strategies are included in *CarromTutor*.

We answered question Q2 above by first identifying that watching an expert's shots, followed by practice, are essential for Carrom mastery. Then we considered a variety of teaching-learning strategies and found Cognitive Apprenticeship [5] to be suitable to form the basis for the instruction design in *CarromTutor*. In traditional apprenticeship, the expert shows the apprentice how to do a task, and the apprentice gains mastery by repeated practice of portions of the task, with diminishing help from the expert. However, masters of a skill often fail to take into account the implicit processes involved in carrying out complex skills when they are teaching novices [5]. Cognitive apprenticeship based tutoring involves designing the instruction such that these tacit processes are brought into the open, where students can observe and practice them. There are many methods associated with Cognitive Apprenticeship, such as modelling, coaching, scaffolding, articulation, and so on. Some of them which relate to providing a chance to students to observe, practice, and learn expert strategies in a domain, are used in *CarromTutor*. These are:

- *Modeling:* In this, a learner observes an expert performing a task with the purpose of understanding it and building a conceptual model of the process in mind. Once the process of achieving a task is clear in learner's mind then he/she practices it. In *CarromTutor* we provide video tutorials and animation (See figure 2) of each Carrom skill to enable the learner to observe the expert performing the task. Videos of Carrom skills are displayed in such a way that working memory of human mind is efficiently utilized by using separate channels of verbal and visual material. We have interleaved textual descriptions of different Carrom skills with its video tutorial to utilize both the channels efficiently. Subsequently, the learner can practice the demonstrated skill (See figure 3) in the exercises associated with the skill. The learner has freedom to place the striker by dragging the mouse, select the coin to be hit by rotating the view, and decide the force of the striker by clicking on the force level indicator.

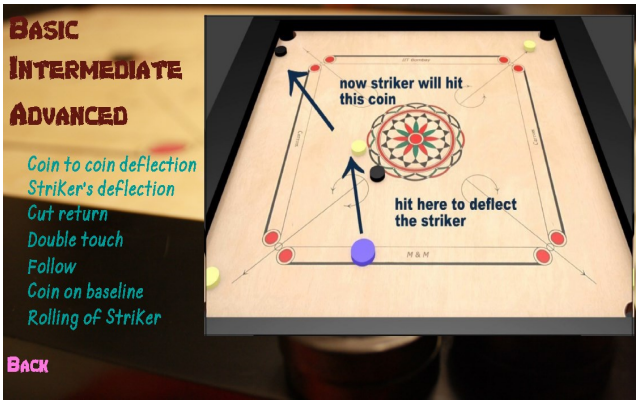

Fig. 2: Tutorial for a Carrom skill

- *Sequencing:* In this, the learning activities are ordered such that the learner acquires expert skills step by step, and also their application in varied contexts. One way

Fig. 3: Learner practicing an exercise

to sequence the learning activities is to arrange them in increasing order of complexity. In *CarromTutor* activities are given to learner such that more and more expert skills are learned gradually, and with increasing level of difficulty in each task. We have categorized the skills into Basic, Intermediate and Advanced. The sequencing of skills of 'Advanced' category can be seen in the left panel of figure 2.

- *Scaffolding:* In this, the teacher provides support to help the learner in performing a task. This support is removed when students are able to perform task on their own. In *CarromTutor* we have implemented rudimentary scaffolding, through providing suggestions or hints to the learner, whenever needed. A screenshot of one such hint is shown in figure 4.

Fig. 4: Example of hints given to learner

We answered Q3 above by first identifying characteristics of educational games [6]. Then we considered a variety of game-based learning strategies [11], [12], [13], and found Van Eck's [13] principles of digital game-based learning environments to be suitable to form the basis for the game design in *CarromTutor*. These are:

- *All learning in games is situated and relevant to the game*. In *CarromTutor*, the real-world scenario is itself presented in the game. For any given situation on the Carrom board, the cognitive aspect of deciding the strategy and choice of a skill, are identical in the real-world and in the game. The animation provided for the shots is also created using realistic parameters. Only the development of dexterity for flicking the striker on a real board is not incorporated in *CarromTutor*.

- *Game playing involves a cycle of interaction and participation, resulting in immersive engagement*. In *CarromTutor*, the game is designed such that a learner to motivated to complete the required objective at the current skill level to make an advancement into the next level. The learner can choose to move his/her viewpoint and see the board from different angles seamlessly. The learner can position the striker at will, flick it in a desired direction and control the force of the striker's hit, similar to a real Carrom game. The simulation of the coin movement is also made realistic by appropriate settings for the material, friction, collision and bounce parameters in the Blender Game Engine. This results in immersive engagement for the learner.

- *Games employ problem-based learning structures*. In *CarromTutor*, problems (exercises) are ordered so that the learning from solving one problem will assist to solve later problems. Learners master content by applying earlier learned skills while learning new skills and strategies. For example 'normal cut' is a 'Basic skill', where the learner needs to hit a coin at an angle so that the coin moves in desired direction. This skill is applied in 'striker's deflection' which is an 'Advanced skill'. This is illustrated in figure 6, wherein the learner needs to do a 'normal cut' on the white coin in order to 'deflect the striker' towards the black coin at the top of the board.

- *Game environments build in feedback and assessment*. In *CarromTutor*, an exercise has multiple ways of getting to the goal. The learner's choice of skill or strategy to be used is assessed and a score is generated. If the score is not the maximum possible for that exercise, feedback is given to the learner on the alternate skill or strategy to be used. For example, in figure 5, the learner can strike along the red line and hit the black coin near the pocket. Alternately, the learner can strike along the red line shown in figure 6, to hit the upper portion of the white coin and deflect the striker before hitting the black coin near the pocket. The former shot(figure 5) rewards 40 points while the latter one (figure 6) rewards 75. This is because if the learner executes the former shot, then it becomes difficult to subsequently pocket the second black coin, as it's path is obstructed. Whereas in the latter shot, the striker hits the white coin, thereby moving it and clearing the second black coin's path to the top-left pocket. Hence learner's who play the first shot are given feedback and hint as in figure 4.

In the next section we describe the functionality of CarromTutor, from a learner's perspective.

Fig. 5: Complex exercise 1

Fig. 6: Complex exercise 1, alternate shot

IV. CarromTutor: Functionality

The sequence of activities that happen when a learner interacts with *CarromTutor*, i.e., the workflow of the system, is shown in Figure 7.

The first scene of *CarromTutor* provides buttons for starting the tutor and usage instructions. The next scene has buttons for tutorial, complex exercises, strategies and game rules. Of these, tutorial and complex exercises form the main parts of *CarromTutor*, and their workflow is shown in Figure 7. If the learner chooses to start a tutorial (by clicking the corresponding button), then all the Carrom skill names along with their categorization appears on the screen. Now clicking on any skill name displays its tutorial video along with text description embedded in it. After the tutorial video of a skill ends, practice exercises of that particular skill get loaded one after another. Currently we have implemented two practice exercises for each skill. The second practice exercise is presented only after the learner successfully executes the first practice exercise. In practice exercises for advanced skills, if the learner is unable to complete the exercise, hints are given so that the learner can select the right skills to apply. The learner can press 'spacebar' to go back to skill selection scene while doing an exercise, in order to review the tutorial video and then continue with the practice.

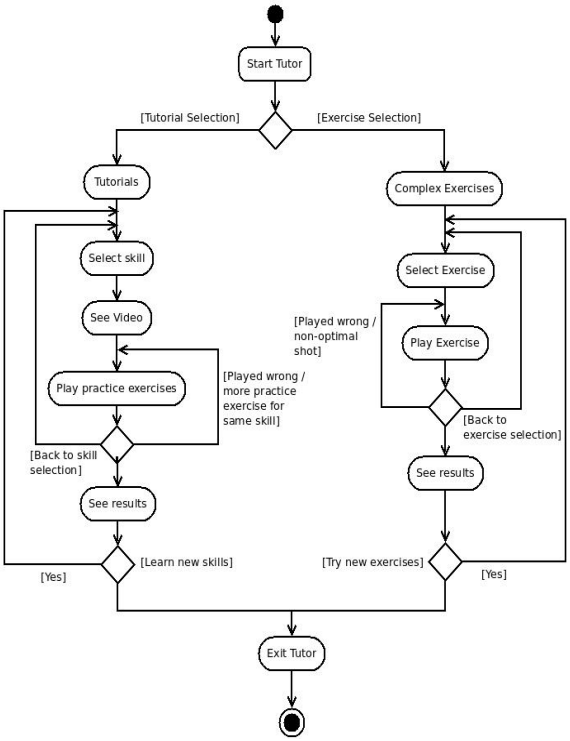

Fig. 7: Learner Activity Diagram

When the learner has completed the tutorials, or feels ready to go directly to complex exercises, the learner can go back to the first screen and then click on the 'complex exercises' button to get a list of such exercises. Complex exercises are comparatively larger exercises than practice exercises. These require two or more successful shots to be performed consecutively. In each of these exercises a complex board situation is presented to the learner. The learner has to think and apply the suitable skills and strategies learned in tutorial section, to pocket all black coins on the board, by playing consecutive shots. *CarromTutor* has a score associated with each shot, which are added together to get the learner's total score for that complex exercise. Now, these complex exercises are designed in such a way that at any point during their execution, there are multiple valid shots that the learner can play. However, all of these shots may not lead to successful completion of that exercise. Moreover, even if a particular sequence of shots leads to successful completion of the exercise, it may not be the optimal sequence for getting the highest possible score for that exercise. The learner can compare his/her score with the maximum score and replay the exercise, i.e., try another sequence of shots, in order to discover the optimal sequence. This enables them to master not only individual skills but also the strategies to be used.

V. Implementation

CarromTutor has been created using *Blender Game Engine* (BGE). Implementation using BGE has three main parts: object modelling, logic editing and python scripting. As shown in figure 8, the modelling area allows the developer to create,

transform and manipulate game object properties efficiently. The logic bricks and python scripting areas are used to implement functionalities in the game.

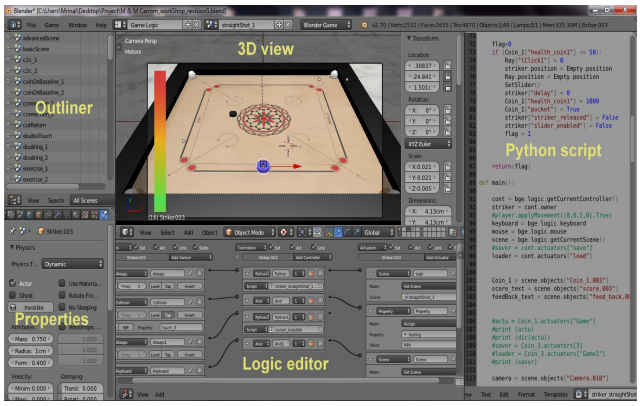

Fig. 8: Blender interface

A. Object modelling

Modelling means creating objects, transforming them according to need, setting up their properties, and so on. BGE provides interactive layers to do modelling. The Carrom board shown in figure 8 is made from cubic mesh objects. Striker and coins are created from cylindrical mesh objects. The carrom board is placed in a room that has been designed by creating plane objects and unwrapping some texture images onto them. The power slider bar, and other objects which are displayed to the screen, have been attached to the camera object so that they can move along with the camera.

The Carrom board surface, border and background, are *static* objects. *Static* objects remain at a fixed position. Moving objects like striker and coins are set as *dynamic* object. BGE simulates the movement of dynamic objects as per the laws of physics. Additionally python scripts can also be used to manipulate their behavior. The developer needs to set the different parameters of these objects so that the object behavior and movement matches with reality. For example, parameters like friction coefficient, damping coefficient, and elasticity, were crucial for real-like visualization of carrom shots in *CarromTutor*. These were chosen by observing an expert play a shot on a real carrom board and then manipulating the various parameter setttings till the same shot was accurately simulated in BGE.

B. Logic editor

The logic editor in BGE is used for scripting to implement the functionalities of the game. It provides *Logic Bricks* to create the desired functionality in the game. These are the rectangular blocks in figure 9. We can set up a collection of logic bricks for each game object. Each object may have different *properties*, such as dynamic, static, rigid body, and so on.

As can be seen in figure 9, *Logic Bricks* have three types: *sensors* (the left column), *controllers* (the middle column), and *actuators* (the right column). The lines are *connectors*, used to establish connections between these three. *Sensors* are related with sensing specific situations or properties. A *sensor* gets triggered when something desirable happens according to the functionalities set into it. When a *sensor* is triggered it sends pulses, which can be set up as continuous mode or instance mode. *Controllers* are required to get the output from *sensors* and feed them either to *actuator(s)* or to a *python* script attached with that *controller*. *Actuators* perform tasks set up by the developer.

For example, in figure 9, 'Always' type sensors (1st and 3rd sensors) are connected with two different 'Python' controllers (1st and 3rd controllers). The sensors triggers after the user plays a shot and the corresponding controllers check conditions associated with the scene after the shot. If the conditions are not met, then the 'Scene' actuator (1st actuator) connected with the controller reloads the same scene.

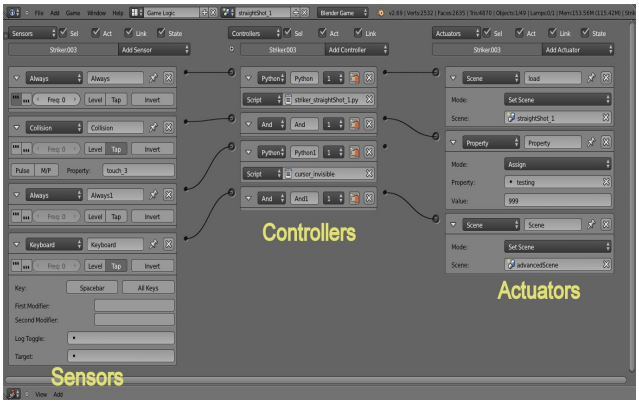

Fig. 9: Logic editor layer in BGE

Implementation of some important functionality of *CarromTutor* using the *Logic Editor* are described below:

- ***User Interface:*** This consists of many scenes created inside the game. Different game objects are needed to create these scenes. Some objects act like buttons in these scenes where users has to click to interact with. In the start scene, there are sensors attached with each objects created for the user to interact with. Mouse clicks on these clickable objects are captured with two sensors of type 'Mouse'. One of them sends positive continuous pulse when mouse is hovering over that object using 'Mouse Over' event while the second one gives a positive pulse instance when mouse is clicked. These two *sensors* along with some 'Property' sensors are attached to an 'And' *controller*. When these *sensors* send positive pulses together to associated *controller* then the corresponding *actuators* executes the function assigned to it. The *actuator* of type 'Action' that is connected with the 'And' *controller*, is responsible for showing an animation created with respect to scaling of the object that it is attached with.
- ***Dependencies between intra-Scene objects:*** Complex functionalities can be performed by establishing connections between logical components of different

objects. If multiple objects are selected together then *Logic Editor* opens them together in a single layer. Now, establishing logical connections between the logical components of theses objects performs the desired dependencies of their actions. For example, videos of five basic skills need to be displayed on one plane. So these six game objects are needed to be inter connected. This is implemented by having one 'Property' sensor from each basic skill being is connected with the appropriate 'Python' controller.

- ***Displaying tutorial videos and presenting practice exercises:*** When the learner selects a skill by mouse click, corresponding tutorial video of that skill is displayed on a plane object which acts as a screen. After a tutorial video is displayed the corresponding practice exercise(s) are presented by setting the appropriate scene of the associated *scene* actuator.
- ***Exercises:*** In practice exercise scenes *Logic Bricks* are used along with striker, coins and various other objects. Some 'Always' type sensors, connected with different 'Python' controllers, are used along with striker objects. Scripts associated with these controllers check if required conditions for that practice exercise are met or not after user plays a shot, and carry out the corresponding actions.

C. Python scripting

BGE provides an interactive layer for creating and editing python scripts. All game object properties and actions can be manipulated using python scripts. Some of these are described below:

- ***Changing user's view according to mouse movements:*** In *CarromTutor* a ***3D*** view is incorporated to offer real life experience while looking at the carrom board from different angles. The learner's view of carrom board changes according to mouse movement. This is implemented by capturing mouse movements and applying them on the camera using the python script, named *"mousemove.py"* [14], downloaded from the Blender community site [15].
- ***Showing striker's path:*** Scripts are used to show the accurate direction of striker movement after it's release. A red line is used to show the striker's path, by having a "Ray" sensor which emits a ray. The direction of the ray is set along the *Y*-axis and the ray is terminated at the first object it collides with.
- ***Striker's orientation:*** Python scripts attached with striker object in each exercise implement moving the striker on base line to take a shot, control the power slider of that exercise, and apply force on striker in chosen direction. User can move the striker on baseline using left and right arrow key of keyboard. To release the striker in proper direction we need to rotate the striker to align with the camera so that striker moves in the direction of the user's sight. While the camera can rotate on three axes, the striker should rotate around Z axis only, and its X and Y axis directions change accordingly. This is implemented using Python API for BGE that provide "orientation" of objects in 3D space.
- ***Striker's force:*** The power slider controls the force with which the striker is released. It is made by using ten mesh objects of type *'plane'*. The height of the power slider is varied by showing different numbers of planes with respect to time. The power slider is attached to the camera so that it moves according to mouse movement.
- ***Executing a shot:*** While executing a shot it is necessary to detect collisions among striker, coins, and border of board. Python scripts are written to detect these collision and do required modifications to various object (coin, striker, border of board) properties. For example, when user pockets a coin, properties of the coin are changed to reflect the same. Other scripts detect whether all coins and striker have stopped moving after a shot, and load the next scene. The collision detection is also used in assessment, determining correctness of user's shot, and points earned.

We have given further detailed description of *Carrom Tutor* implementation in [16] and [17].

VI. Evaluation

We have done preliminary evaluation of *CarromTutor* through a study involving eleven users - five beginners, five intermediate players and one expert. Our research questions (RQs) were:

- *RQ1*: What is the perception of learning due to *CarromTutor*?
- *RQ2*: What is the usability of *CarromTutor*?

We answered these RQs by doing as follows:

- To answer RQ1, we administered a complex exercise from *CarromTutor* as a pre-test, had users interact with *CarromTutor* for two hours going through all tutorials and practice exercises, and administered a similar complex exercise as a post-test. We collected data of user's perception of learning, through a survey having questions: (i) How many new carrom skills have you learned?, (ii) How easy was it to complete the complex exercises before interacting with *CarromTutor*? (iii) How easy was it to complete the complex exercises after interacting with *CarromTutor*?, and (iv) How easy was it to relate the tutorials with the shots required for complex exercises?
- To answer RQ2, we administered the *System Usability Scale* (Sus) scale *SUS* and analyzed the user's responses to the questions. The SUS statements about complexity of the system, integrity of functions, inconsistencies, user's intent to use the system again, need of technical support, and so on. The participants respond to these statements on a 5-point Likert scale ranging from 1 (strongly disagree), to 5 (strongly agree). *SUS* provides reliable results ([9]) and can do so even for a small set of users.

- In addition, we also asked users to informally compare the learning, usability and interactivity experience of *CarromTutor* with an online game *Carrom King*. To avoid biasing these responses, we had half the users play *Carrom King* before interacting with *CarromTutor*, while the others interacted with *CarromTutor* before playing *Carrom King*.

We analyzed the data and responses to answer the RQs. These results are reported next section.

VII. Results

The results for survey question (i), in *RQ1*, i.e., the self-reported number of new carrom skills learned by different categories of learners is given below in tabular form:

RQ1	Beginner	Intermediate	Expert
New Carrom skills learned (Avg.)	4-5	5-6	3-4

The results for survey questions (ii), (iii) and (iv), in *RQ1*, are shown in table given below:

RQ1	Easy	Medium	Hard
Ease of completing complex exercises before *CarromTutor*	0-0-0	0-1-0	5-4-1
Ease of completing complex exercises after *CarromTutor*	5-4-1	0-1-0	0-0-0
Ease of relating tutorials with shots required for complex exercises?	4-3-1	1-2-0	0-0-0

In this table each cell has three values. These correspond to the number of beginners, intermediate players and experts in that cell, respectively. For example, in row "Ease of completing complex exercises after *CarromTutor*", column "Easy", the cell has the values 5-4-1. This indicates that 5 beginners, 4 intermediates and 1 expert, reported that it was easy to complete the exercises after interacting with *CarromTutor*. The values in other cells are to be interpreted in like manner.

It is encouraging to note that all the participants who felt the exercises to be 'Hard' before interacting with *CarromTutor*, subsequently felt similar exercises to be 'Easy' after interacting with *CarromTutor*. We have observed from learner's moves while playing exercises that they were able to identify the skills which are required in exercises after interacting with *CarromTutor*. Also, most of the participants reported it 'Easy' to relate the tutorials with the exercises. Since the total number of participants in the study is small, we did not convert these numbers into percentages or carry out further statistical analysis.

The results for *RQ2*, are shown in Figure 10 below. On the X-axis are statements from the SUS scale and on the Y-axis are the average ratings for each statement, where individual ratings can range from 1 (strongly disagree) to 5 (strongly agree), on a 5-point Likert scale. For quick reference, we list the SUS scale items [8] below:

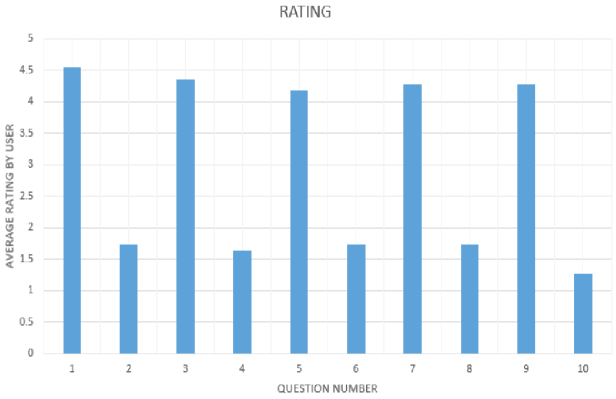

Fig. 10: SUS analysis of *CarromTutor*

1) I think that I would like to use this system frequently.
2) I found the system unnecessarily complex.
3) I thought the system was easy to use.
4) I think that I would need the support of a technical person to be able to use this system.
5) I found the various functions in this system were well integrated.
6) I thought there was too much inconsistency in this system.
7) I would imagine that most people would learn to use this system very quickly.
8) I found the system very cumbersome to use.
9) I felt very confident using the system.
10) I needed to learn a lot of things before I could get going with this system.

The odd-numbered statements are phrased in 'positive' terms, while the even-numbered statements are phrased in 'negative' terms. The results show that most of the participants either "agreed" or "strongly agreed" with the 'positive' statements, while they either "disagreed" or "strongly disagreed" with the 'negative' statements. These ratings were converted into a SUS score as per the standard formula [8]. A SUS score of 68 is considered as average. The SUS score for our *CarromTutor* is 84.09 (grade A, as per [8]).

We also sought open-ended feedback from the participants. Some of their statements are quoted (verbatim) below:

- Very interactive tutoring system.
- Presence of 3D/user view is an added advantage.
- Feel of a real Carrom board while playing the exercises.
- Rotation of striker with mouse movement is very good.
- Line to draw striker's path is very useful for smooth handling of striker.
- Other applications does not provide any tutoring so this tutor encourages learning.

Two comments from participants comparing *CarromTutor* with *Carrom King* are quoted below:

- "*Other app (Carrom King) doesn't have a good interactive experience; lacks 3D point of view; doesn't provide any learning as such; doesn't encourage me to know more about the game, rather discourages to a certain extent. Your system (CarromTutor) scores well on all these points*".
- "*3D view helps in relating the game to the actual one. Another game lacks this approach. In the other game, there are no tutorials and to improve the skills, you need to spend more time playing the game. The line for the striker's direction helps to select the striker's position. Other game lacks this facility*".

VIII. Discussion

The answers to our research questions are as follows:

For *RQ1* - What is the perception of learning due to *CarromTutor*? - we found that all categories of learners reported learning of 3-6 new carrom skills in their first interaction with *CarromTutor*. Ten out of the eleven learners perceived that it was 'Hard' to complete the complex exercises initially, but it was 'Easy' to complete these exercises after undergoing the training in *CarromTutor*. It was also observed that before interacting with *CarromTutor* learners were unable to apply the required carrom skills even after several trials. But this situation dramatically alters after learners experiences *CarromTutor*.

For *RQ2*: What is the usability of *CarromTutor*? - we found that the participants usability ratings on the SUS scale resulted in a SUS score 84.09, thereby categorizing *CarromTutor* as a grade A system. The open-ended feedback from the participants triangulated well with the SUS responses, thereby confirming the usability of *CarromTutor*.

The feedback on comparison of *CarromTutor* with a popular online game was as expected, and confirmed the usefulness of incorporating 3D views, realistic simulation of movement, tutorials and exercises. The feedback on the tutoring and game aspects also confirmed that our design of applying principles from cognitive apprenticeship and game-based learning are effective. We believe that our approach for the design of *CarromTutor* may be generalized and applied to build tutoring systems for other games as well.

One limitation of our study is that we only have a preliminary evaluation of learning and usability. Hence these results may not be statistically significant. More experiments and in-depth research studies are required in order to examine generalizability of our results.

IX. Conclusion

Many online carrom applications provide a game environment for playing carrom with human or computer opponent. While *CarromTutor* does not provide the feature of playing with an opponent, it explicitly addresses the teaching-learning of carrom. It systematically incorporates principles of cognitive apprenticeship and game-based learning, and provides an opportunity to learn carrom skills and strategies in a 3D game environment, that is close to a real-life scenario. To the best of our knowledge *CarromTutor* is the first system to focus on teaching-learning of carrom, rather than playing of carrom.

CarromTutor was built using Blender Game Engine in order to simulate a realistic setting. Its parameters allow learners to not only control their views and actions, but also see the effect of their actions as they might occur in a real game. A user study of the learning and usability of *CarromTutor* confirmed that it is an effective and attractive system for learning carrom skills and strategies.

While further experiments are required to confirm generalizability of specific results, we believe that our approach of incorporating principles of cognitive apprenticeship and game-based learning would be applicable for the design of tutoring systems for other games as well.

Acknowledgment

The authors would like to thank Rwitajit Majumdar, Shitanshu Mishra, Sameer Sahasrabudhe and Nitin Ayer for their help. The authors would also like to thank those who participated in *CarromTutor*'s evaluation study.

References

[1] Wikipedia *Carrom*. http://en.wikipedia.org/wiki/Carrom.

[2] Carrom King. http://twoplayergames.org/play/719-Carrom_King.html.

[3] U.S.Somanchi, *Carrom play techniques*. http://uscarrom.org/docs/CarromPlayTechniques.pdf.

[4] Tracey Hall. *Explicit Instruction*. National Center on Accessing the Curriculum, 2002. http://aim.cast.org/learn/historyarchive/backgroundpapers/explicit_instruction.

[5] Allan Collins, *Cognitive Apprenticeship*, chapter 4, Handbook of the Learning Sciences. Cambridge Univ. Press, 2006.

[6] Imgrad Schinnerl, Maja Pivec, Olga Dziabenko, *Aspects of game-based learning*. In Proceedings of I-KNOW '03, Graz, Austria, pages 6–20, July 2003.

[7] Blender Game Engine. http://blender.org.

[8] Sus analysis application. http://www.mohini.0fees.net/iit/welcome.php.

[9] Measuring usability. http://measuringusability.com/sus.php.

[10] Tom O'Donnell. *Chess Teaching Manual*. Chess Federation of Canada, 1997.

[11] J. Gee, *What Video Games Have to Teach Us About Learning and Literacy*. Palgrave Macmillan, 2nd edition, December 2007.

[12] Marc Prensky, *Digital game-based learning*. Paragon House, 2007.

[13] Van Eck, R. *Building Artificially Intelligent Learning Games. In Gibson, D., Aldrich, C., Prensky, M., Games and Simulations in Online Learning: Research and Development Frameworks (pp. 271-306)*. Idea Group Inc., 2007.

[14] Riyuzakisan. *3d art - game design*. http://riyuzakisan.weebly.com.

[15] Blender community, http://blenderartists.org.

[16] Mrinal Malick. *Carrom Tutor : Game-based Learning and Implementation*. Master's thesis, IIT Bombay, June 2014.

[17] Mayur Katke. *Carrom tutor : Playing strategies and impementation*. Master's thesis, IIT Bombay, June 2014.