Project Part 2: Design Alternatives to an Automatic Competitive Straight Pool Scoring System

Introduction

The goal of our system is to alleviate the typical pool player from having to keep score manually, for example during local league play. To accomplish this automation, we intend to place a video camera to observe the pool game and provide an interface to track each player's scores. An example story of the high level player experience while using our system now follows. The players walk up to an empty table already configured with the visual scoring system, and they decide how they want to play. One player initializes the game and the interface shows that a new game has begun. Player one begins by breaking and play commences with players alternating turns while the system automatically keeps track. Scores are automatically updated on the interface and players are free at any time to adjust the scores to correct for errors should they occur. The system displays that a player has won and the players can then review their game. Finally, the players can clear the game and leave to continue their lives.

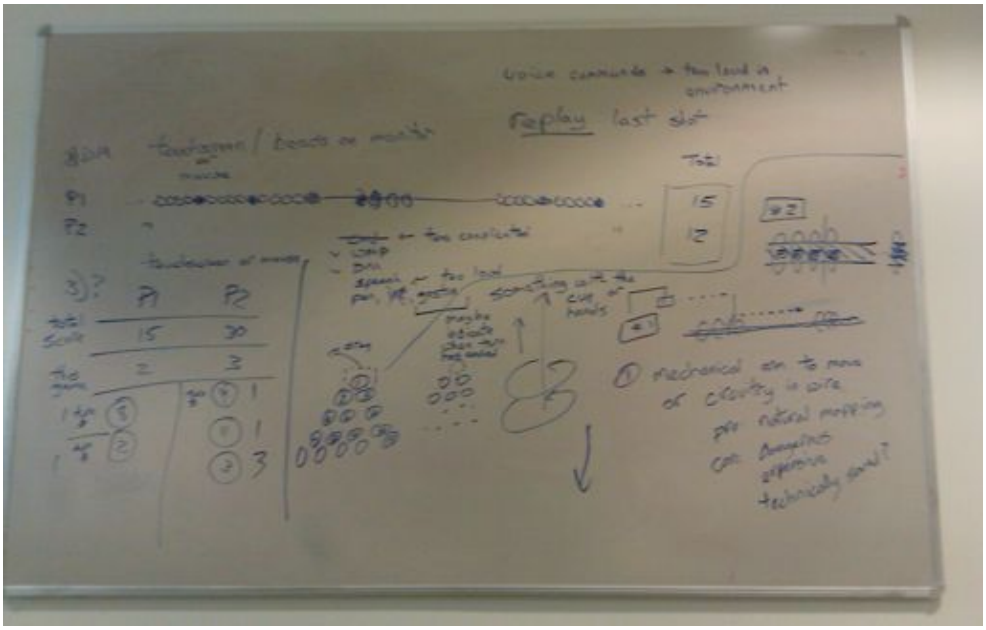

Figure 1: Team whiteboard brainstorming session

The interface to use can range from the classical mechanical dials of early manual scoring systems or could be the latest speech interface, touch screen, or AR design. This interface may be on a computer's screen allowing use of a mouse, keyboard, or other input device; or the interface may be a construction of something more or less like a personal computer. Our own ideas, system usability, and feasibility within our resource constraints will define how our project can go forward. An initial

brainstorming session carried out on a whiteboard shown in Figure 1 resulted in ideas about string-of-beads metaphors. We thought of this metaphor each on a touch screen and in a physically realistic recreation of manual beads. But the latter idea turned into a different manual mechanism. As well, virtual balls seemed a natural mapping for visual score tracking. As well, we realized that an instant replay feature would be very desirable for users as well as relatively easy to implement given that we intend our system will run with a webcam at all times. We eliminated the possibility of voice commands because of the background noise likely present in a typical pool hall. What follows are three more detailed explorations of potential system interfaces spawned from the brainstorming session for our automated scoring system.

Evaluation of Design Alternatives

Design 1: Automatic Table-counter Scoring

This is the only proposed system that does not use a screen-based interface to provide user feedback. Instead, this system uses scoring mechanisms (similar to air hockey tables) which are build directly into the table, which is shown in Figure 2. As each player pockets balls, the table counters will be automatically updated to reflect the new score. In the event that the system has made a mistake, manual adjustment of the table counters may be performed to correct the system.

In this design a small, simple console is embedded into the head rail of the table. Sensors are placed within each pocket to detect when a ball has been pocketed. The system consists of three buttons: a reset button and a toggle button for each player. There are also two 2-digit table counters for each player with dials located next to each digit for manual adjustment. When a ball is pocketed, the system automatically increments the current player's score by one. Similar to chess timers, upon approaching the table the current player will press the appropriate "turn button" (depending if the player is P1 or P2) which will automatically cause the opponents turn button to release. The turn button indicates which player should be given points as the sensors detect pocketed balls. The reset button is used to clear all system state to start a new game. For this design, clearing system state simply results in both 2-digit counters being reset to zero. Additionally, the system also allows for manual adjustment in the event of an error. System correction simply involves the rotation of the dial(s) to the correct score. This design leverages the existing table counter method discussed for scoring straight pool discussed in the first part of the project.

Pocket sensors connect to the embedded table-counter module by routing thin wires either around the edges of the table, or within the rails themselves. A variety of electromechanical sensors could be utilized to detect pocketed balls. One such sensor is a small flipper placed inside each pocket, where incoming object balls will cause the sensor to be triggered. With this scheme, players may need to ensure that sensor readings are not hindered by too many object balls within a pocket. However, this should not be inconvenient to users, because players often rearrange balls to keep pockets from getting too full. Using this design, the entire system may be constructed mechanically, or by processing sensor data within a small embedded device within the counter module.

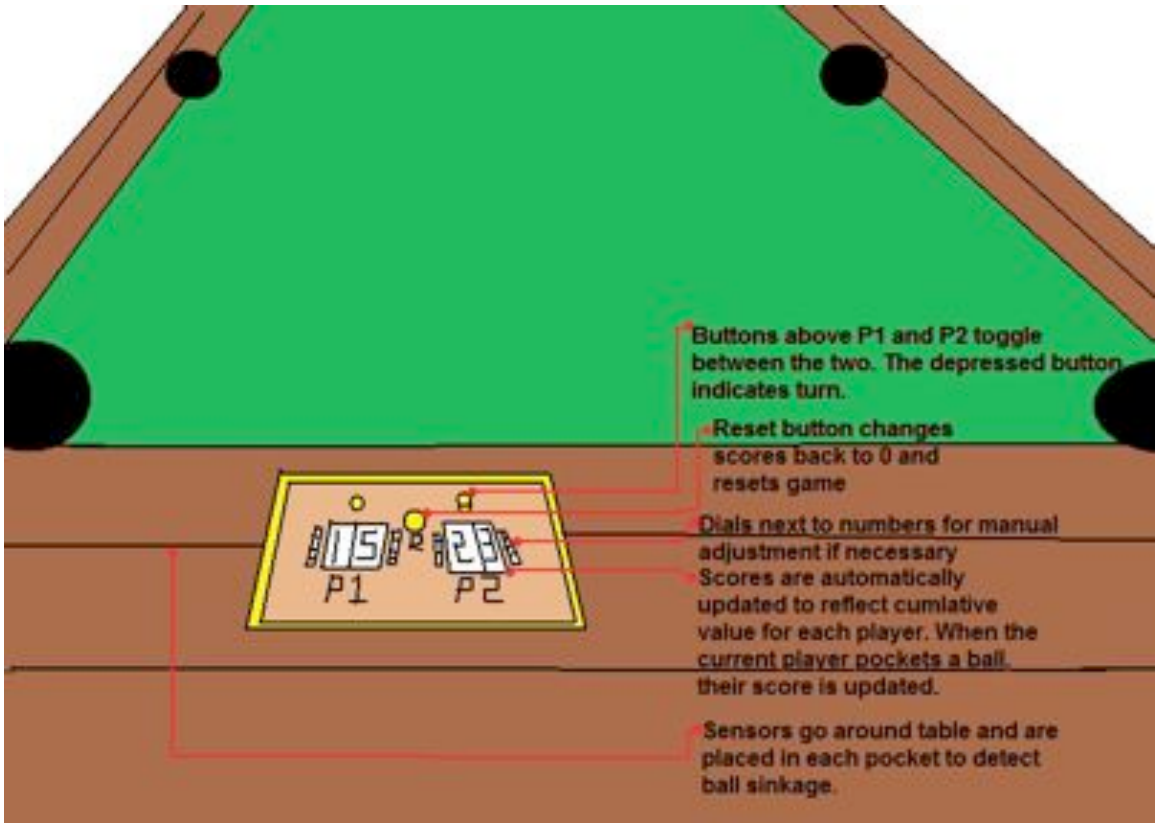

Figure 2: Sensor data is used to update table counter embedded within the head rail

Because the table-counter method of scoring is familiar to many straight pool players, this system should be quickly learned by new users. When a player's turn button is pressed down, the opponent's turn button should pop up. Since one button is always up and the other down, it should be immediately clear how these work, especially to anyone who has seen a chess timer. The reset button could either say "R" or "Reset." In either case, it's function should be immediately clear once it is pressed by the user (hopefully not in the middle of a match). The dials next to the numbers are the same as on any table-counting scheme, and there should be no confusion there. The only remaining issue is alerting a new user to the existence to the automated scorer. The wire running around the table should be the first affordance indicating that something has been added. Players will also quickly notice the flippers inside each hole. They'll instinctively push one down, and seeing the numbers change should provide instant feedback on how the system works. A useful addition is to make sure the dials make some sort of sound (perhaps a subtle clicking noise) that should let the user know when they are turning, thereby providing audio feedback. Also, since this system limits scores to two digits, players may be inclined to only play games up to 99. However, the system should scroll back to "00" when a ball is pocketed after 99, which then relies upon the users to remember how many times the counter has overflowed. Although a third wheel could be added to allow games greater than 99, its inclusion complicates the layout of the manual dials as well as diverges from the standard 2-digit table counter.

This design follows a minimalist philosophy of "less is more," and is a slight improvement over existing systems. This design should require less interaction than standard physical systems (assuming players legally pocket balls more often than

they scratch), and the act of simply pressing a turn button prior to shooting requires significantly less user effort than constant table counter adjustment. In a best-case scenario, the proposed system should noticeably speed up the game pace.

While this system provides a moderate level of convenience to end users, it is really not a practical design. The cost of purchasing the system combined with the pool table modifications necessary for installation may outweigh the benefits. In addition, this system does not do enough to reach our goal of making the entire experience of pool league participation easier. Although this design may help alleviate the scoring of individual games, but it would not ease the process of bookkeeping across multiple games. Also, since players cannot enter custom names for use by the system, they must remember if they are P1 or P2 throughout the game.

Although the system has some minor advantages, the scoring accuracy could be a major concern in this design. The system does have the potential to make mistakes (particularly on scratches) and if any mistakes go unnoticed for a few minutes, they may be very difficult to correct. Example: Let's say someone accidentally touched a sensor and they were awarded an extra point, but it went unnoticed. Then, a few minutes later, one of the players notices that the score does not add up correctly (the combined score is greater than the total number of balls pocketed). In this situation, it is very difficult to determine which player has been awarded too many points, and the system will provide no hints to identify where a mistake has been made. However, similar accuracy difficulties are also found within the manual scoring of straight pool.

Potential Login and Setup Procedures for Designs 2 and 3

Once users approach the system, they will be presented with a login/setup interface that is used to define general player and game attributes. The login/setup interface will provide the system with information regarding the players' names as well as the game count objective. This interface can be used for any of the three proposed designs to obtain the relevant information.

The first screen that will be shown is a *login screen* that tells users the purpose of the system. If the system were to be commercial, the *login screen* might allow users to login with system accounts which can be used to automatically retrieve the user name and other prior statistics. Although this feature will not be considered in the initial usability prototype, the login screen can later be modified to include more advanced features.

The user presses start (either through a mouse or a touch screen interface) and a name entry screen will appear. By default, players names automatically entered into the text boxes will be Player1 and Player2. If a user wishes to change their name, they will select the text field and change their name accordingly. If a touchscreen interface is incorporated into the design, a qwerty-style keyboard will appear for the user entry. If a player wishes to change the name used by the system, they will be able to enter a name into a text box. By allowing users to enter specific names into the system, they will not be required to remember if they are Player1 or Player2. After the user has entered in a name (or left the default name within the text field), they are presented with three possible buttons: QUIT, BACK, and OK. Throughout

the menu system these buttons have persistent functionality: QUIT cancels the current game, BACK returns to the previous screen in sequence, and OK continues to the next screen in sequence. Because quitting will result in data loss for the entire game, including name information and the game objective, a pop-up menu or alternate screen will be presented. At this *quit screen*, confirm quitting with OK or return with BACK.

Although the functionality obtained through the QUIT button can be accomplished by hitting BACK until *login screen* is reached, this serves as a shortcut for players who wish to clear all system data, or new players who wish to take over a previously abandoned session.

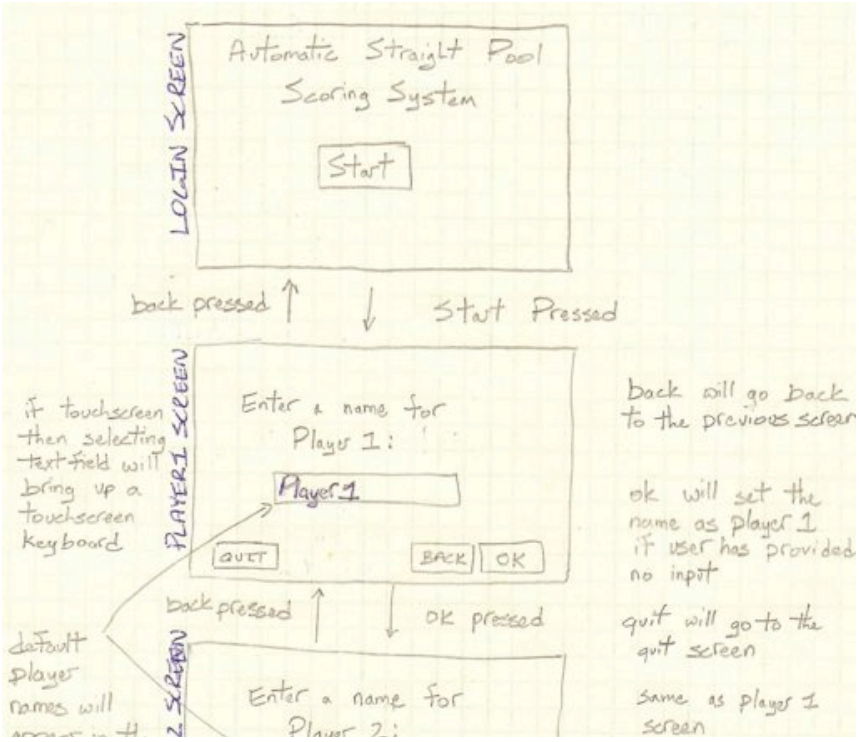

Figure 3: System login and player1 name selection screens

After entering in their names, the users will arrive at the *objective screen* to define the current game's objective. Straight pool games may be played with any number of balls as the game objective. The first player to earn the number of predetermined objective points wins the game.

Also on the *objective screen* is a handicap parameter. If two players of unequal skill are playing one another, then a handicap system may be incorporated to "level the playing field." This screen will present users with the option of using a handicap (in the form of radio buttons). If a handicap is not desired, then only the text box at the top can be used for data entry. The text fields at the bottom will be disabled or "grayed out" in this case. The number entered at the top will be used to populate both boxes at the bottom. The player name labels (i.e. <Player1> and <Player2>) will be populated with the names entered by the users at the previous screens. If either user has entered a name incorrectly, they may go back to the previous menus by using the BACK button to make name corrections. If the users wish to play a

handicapped game, they may select "Yes" for the Use Handicap option. In this setting, the text box and labels on the top are disabled, and the text boxes at the bottom are enabled. By default, 100 object balls--a commonly-used amount--will be set for the game objective.

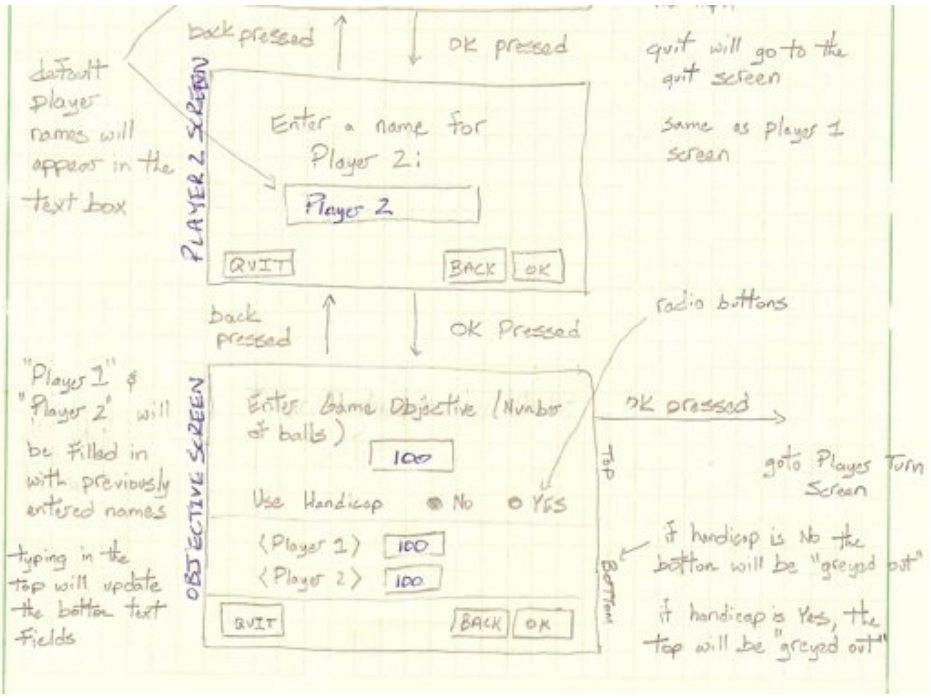

Figure 4: System player2 name selection and game objective screens

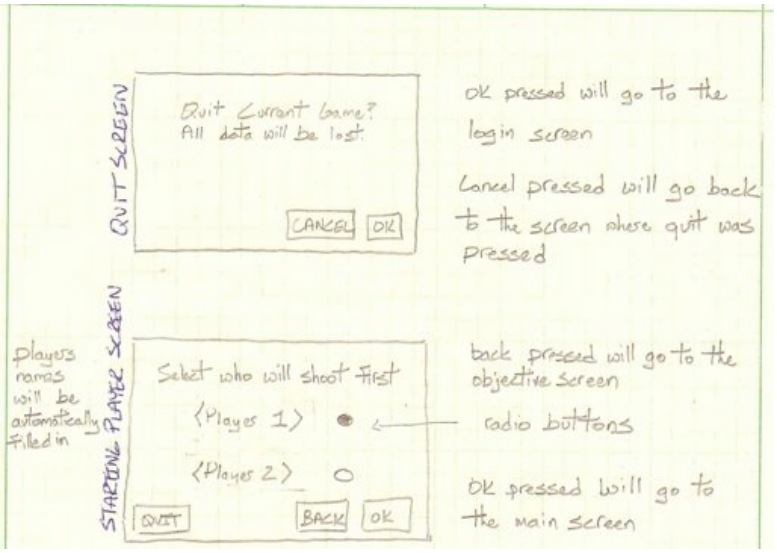

Figure 5: System Quit and Starting Player Screens

After then selecting OK from the *objective screen*, the user will be taken to the *starting player screen*. As in previous screens, the user-entered names will appear in lieu of Player1 or Player2. Here, players will lag or flip a coin to determine which

player shoots first. At the *starting player screen* the players will determine via radio buttons who shoots first. Then selecting OK will cause the system to progress to the *main screen*.

Design 2: Direct Manipulation Bead Scoring System

The idea of this design is to be as simple as possible for the user, while translating the existing bead scoring metaphor into a user interface based on a touch screen and direct manipulation. This design incorporates the use of a webcam to observe table events and computational device(s) for signal processing and system interface display.

One low-cost method to interface webcam data and a computer is to use a traditional screen display, but users may not be comfortable with using a computer within a pool hall. Users will certainly not want to have a command line interface or to have to use the keyboard as main input for the system. Although a mouse input may be slightly better than a keyboard for system interaction, a touchscreen would be ideal. In fact, touchscreen games are widespread in pool halls, so while using a traditional computer may be foreign, a touchscreen interface will appear familiar. If we want our interface to be user friendly, elements such as deep menus and window multitasking should be avoided. Therefore, to create this simple interface, we use a full screen display with only buttons and a direct manipulation dialog to interact with system state.

In order to provide a good conceptual model which would be immediately recognized by the user, one option would be to use preexisting score-keeping metaphors. Thinking in this vein, we got the idea of creating an interface representing the physical string and beads that is commonly used for scoring straight pool.

The sketch in Figure 6 shows the touchscreen interface. Both players scores are displayed using two representations: string-and-beads (from the physical model) and numerical values of the score. Beads on the left hand side of the string represent points that have already been earned, while beads on the right represent remaining points needed to win. At the beginning of a match, all beads will be on the right hand side and the numerical score will be set to zero. The scores of both players are displayed at the same time and the name allows the users to know whose score is on the top and bottom. The goal is given to the user both numerically as well as the remaining beads shown on the screen. The computer also displays a *now-playing* field where the name of the current player shooting appears. A button "switch player" allows player control over changing of turn if, for example, the system missed a turn ending event. A QUIT button also allows the users to stop the game at any point to return to the main menu. A pop-up menu should appear to confirm this action because it will likely be irreversible.

The computer displays scores very simply: every time a player's turn ends, the computer updates scores both in the number text field and by moving the appropriate number of beads from right to left. Correcting the computer will follow a natural mapping: beads are slid from one side to the other to adjust the score. The numerical score will be updated as soon as the beads are moved (even if the user has not yet released the beads) to give immediate feedback. Several beads can be dragged at once by dragging one in the middle of one side. This is just the same way it would be done physically. For example, if you want to give Player 1 five

points as depicted in Figure 7, you would drag the fifth bead of the right side to the left side. This bead would "push" the other beads along with it to the left side.

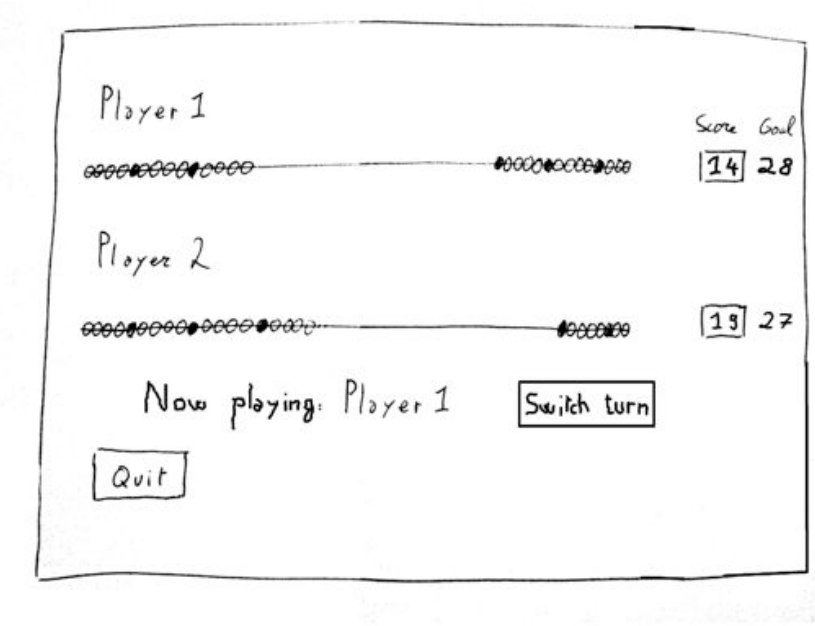

Figure 6: User interface design of the string of beads option

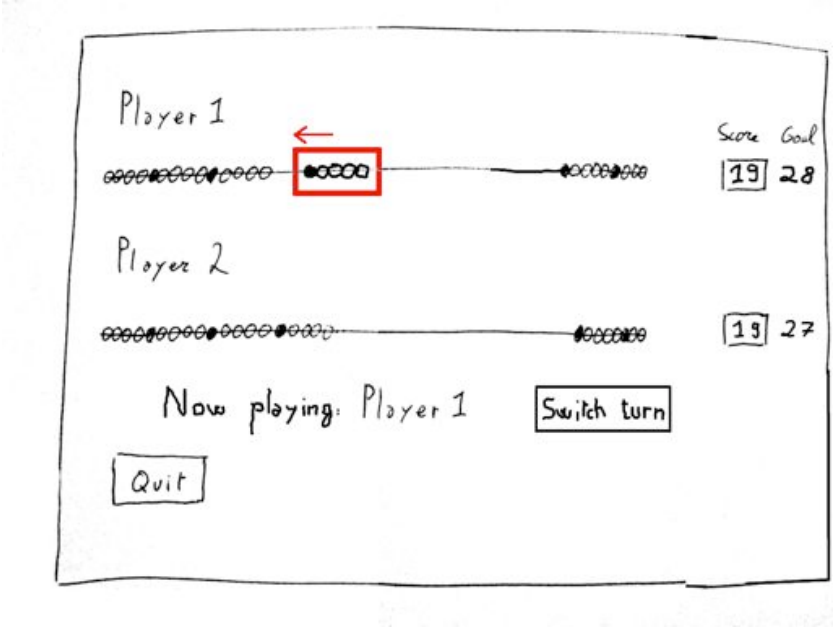

Figure 7: Sliding beads for this design

To make sure that the user is able to verify that the computer is not making mistakes, the computer should move the beads slowly so that the user can verify the number of points he is getting for this play. While a user is playing, the computer should shift a smaller group of beads representing the current score from the right

side to the middle area of the string in order to provide sufficient feedback about the way it keeps score (blue square in Figure 8).

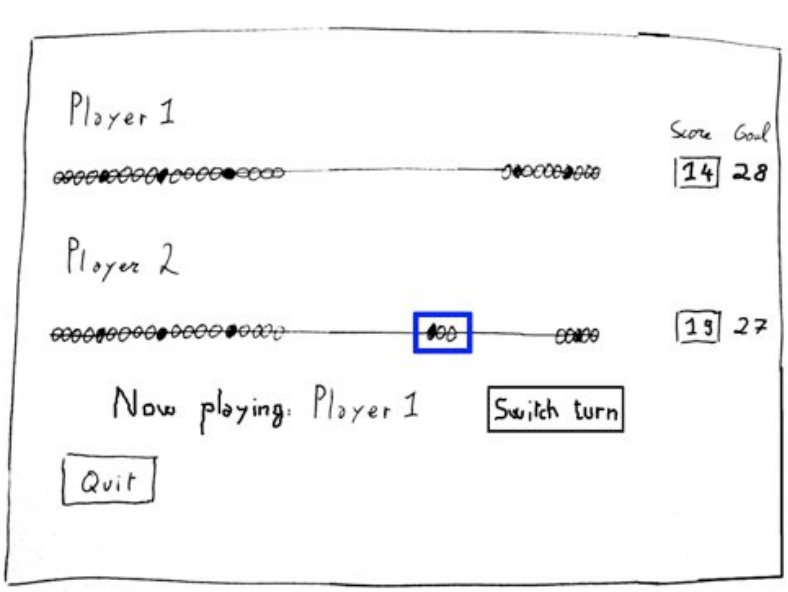

Figure 8: Computer indication about current scoring

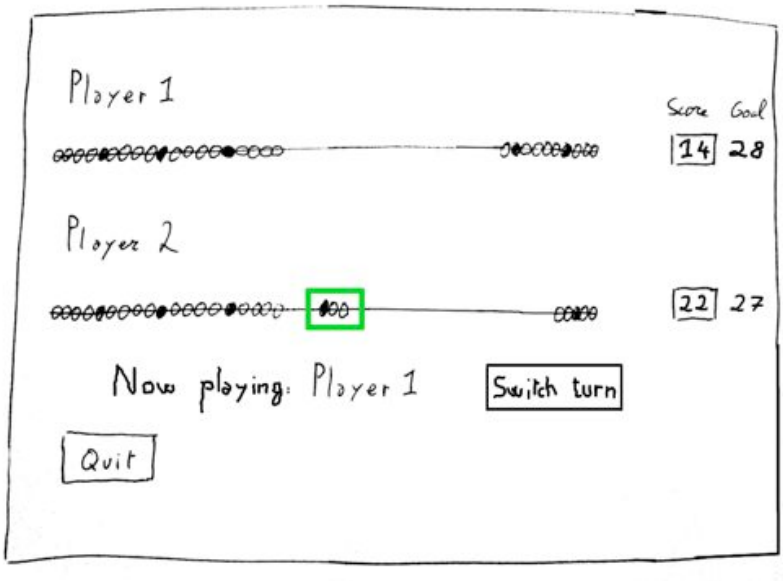

Figure 9: Computer indication about last turn score

After the end of the turn, the computer will move the group of beads to the left side but keep them shifted a bit so that the users have time to look at the number of points changing (green square in Figure 9). Once the next turn has started, the beads will be completely shifted to the left.

This interface has the advantage of being usable with both a touchscreen or a mouse, although as we mentioned in P1, a touchscreen would be more convenient for the user. Another good point is that it is really easy to use. Any pool player can figure how to use the beads if their appearance is similar enough to the physical. It is very user-friendly and in particular for people who do not like computer as they do not need to have any prior knowledge or specific understanding to use this interface. As there are not many commands available, the user will learn to use the system very quickly. As mentioned before, the conceptual model is good thanks to the direct mapping with the physical system. Lastly it allows the user to know at a glance who is leading by looking at the beads and does not require to do the math to know the difference of score (looking at the difference of beads stack is more convenient than doing the subtraction). As well it displays the numerical score at any time of the game so that the user does not need to count the beads.

However it has a lot of flaws, the first one would be that it does not take advantage of the touchscreen - computer - webcam capabilities. The minimalist interface does not provide any improvement except the automatic scoring although a lot of processing power is available. Therefore the user might feel frustrated that the computer does not display more information about its internal state (how it keeps track of the score) which gives it a limited visibility. The user may also regret that it does not provide more advanced technologies which are usually associated with computers such as statistics. And for high-scoring games, the number of beads required to represent each point would necessitate them becoming quite small and hard to manipulate with finger selection. This could be solved by making different types of beads but that makes things more confusing, break the natural mapping to traditional systems, and departs from the theme of simplicity.

Design 3: Play History with Direct Manipulation

A third design option, also based upon a direct manipulation interface, is shown in Figure 10. This design relies upon the use of a webcam attached to a lighting fixture located immediately above the table. Video captured through the webcam will be sent to a computational device for processing, and then statistics/scoring information will cause the GUI to be updated.

In this design, there are four rows for each player, with one player assigned to each main column. The names entered by each player in the login/setup process will be used for display in the main screen. The *Rack Points* and *Balls this Rack* rows are used to indicate the points that have been earned by each player for the current rack. Many straight pool players isolate the points earned in the current rack before aggregating total points (this assumes a traditional bead style counting scheme) in order to assure that all 15 balls have been correctly scored. This design leverages that common practice to help provide visibility of system state to the users, and is explained in detail below. The *Total* row is used to provide a combined score, which sums all points made in all previous racks as well as all points that have been earned in the current rack. Scores shown in this row represent the total current score by an individual player. The *Goal* row is used to indicate the number of points that must be made in order for a player to win the game. Both players will typically have the same goal during a match, but may differ if handicaps are used. The *Goal* row will be displayed for both handicapped and non-handicapped games in order to facilitate a consistent model for both operational modes. At the bottom, there are four buttons: QUIT, STATS, REPLAY, and NEXT RACK. The QUIT button offers the same

functionality as described in the initial setup/login phase. The STATS button will allow players to review game play statistics, which may be viewed at any point during or after the game. The REPLAY button is used to take users to a screen to replay the last shot as recorded by the system. Finally, the NEXT RACK button is used to clear data from the *Rack Points* and *Balls this Rack* rows. After both players agree upon the scoring data and all 15 balls have been accounted for, the players' scores within the total cells will be accurate. Because re-racking the balls already serves to naturally delay game play, the users' action of pressing this button should have minimal impact. The system could also detect when a re-rack is occurring, and perform the NEXT RACK action autonomously. This interface also includes a player turn indicator that consists of an arrow and rectangle that encircles the player currently shooting at the table.

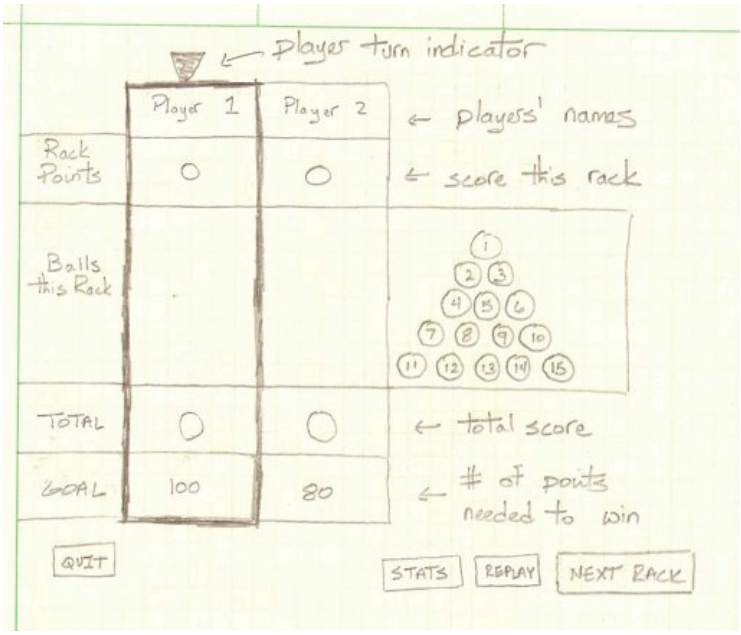

Figure 10: Overview of Main System Screen

During a match, players will pocket balls which will be interpreted by the system and scored to the appropriate player. Once the shooting player pockets a ball, the *Rack Points*, *Balls this Rack*, and *Total* fields will be updated accordingly. This will typically result with the numbers located within the Rack Points and Total fields to be incremented by one. In addition, once a player has legally pocketed a ball, the corresponding object ball will move from the *virtual rack* (located on the right) into the player's area. All balls that have moved from the virtual rack will be "greyed out" so players can easily determine which balls from the current rack have already been pocketed. The purpose of having this information listed in more than one area is to provide maximum feedback and visibility to the users. From this information, players can easily determine how many balls they have pocketed in the current rack, as well as the specific balls they have pocketed. In the event that the system has made a mistake in the scoring of a shot, users can easily correct system state by dragging virtual balls to/from the virtual rack and to/from other players. Figure 11 shows an example of how virtual balls will be moved during game play. On player1's first turn this rack, the user has pocketed the 1, 5, and 12 object balls. After each completed shot, the balls move from the virtual rack into the player's area. After player1 misses a shot, the system recognizes that a miss has occurred and transfers

the turn indicator to player2. Player2 then pockets the 13 ball, and continues to shoot until the user misses. If player2 actually pocketed the 9-ball, but the system mistakenly believed the 13-ball was pocketed, the user could easily fix this situation by dragging the 9-ball from the virtual rack to the user's cell, and then by dragging the 13-ball back to the virtual rack.

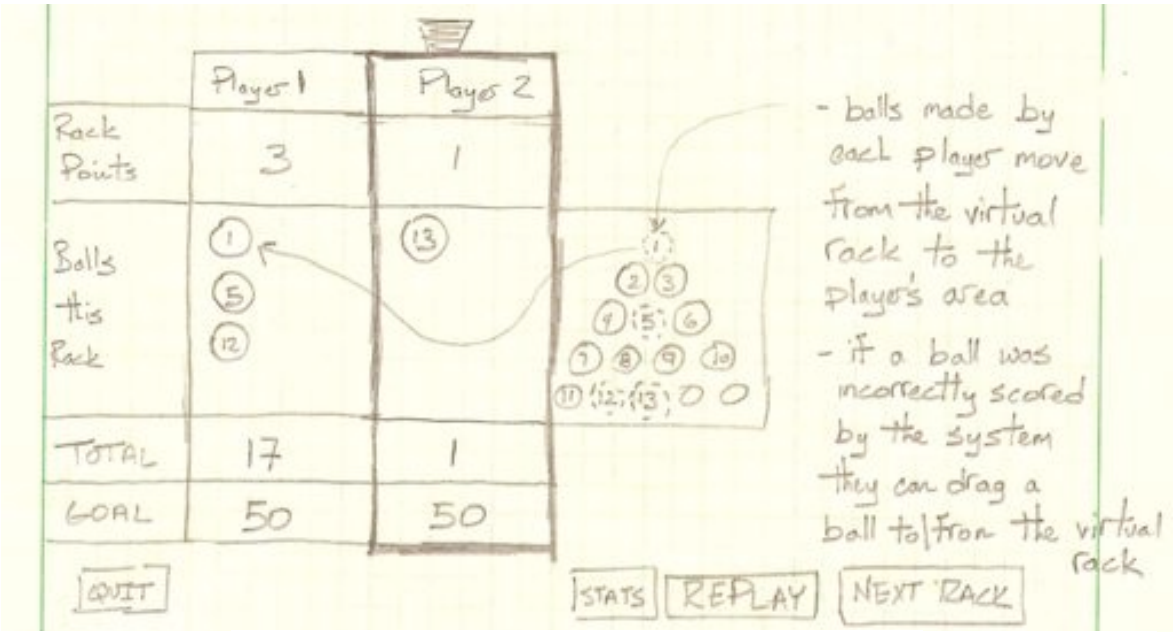

Figure 11: Direct Manipulation to Change the Score

During play, the system could mistakenly detect or reject a cue-ball scratch (or any other fault/event that causes the turn to transfer to the opponent). When this happens, the turn indicator will incorrectly shift to the opposing player. In order to remedy system state, users will have the ability to manually drag the turn indicator to the appropriate player. Figure 12 shows an example of where the system mistakenly changed turns to player1 when player2's turn was not over.

In this design, one of the major goals is to develop an "intelligent" system that can correctly score events in real-time from the webcam data, while still allowing users to override any and all system state. Direct manipulation of the virtual balls can be used to change scores within the current rack. Additionally, mechanisms will be provided to change the total score and even handicaps in the middle of a game. The system should NOT dictate the actions that are allowable to the users. The inclusion of such mechanisms would only serve to frustrate the end users, and would diminish the usability of the proposed device.

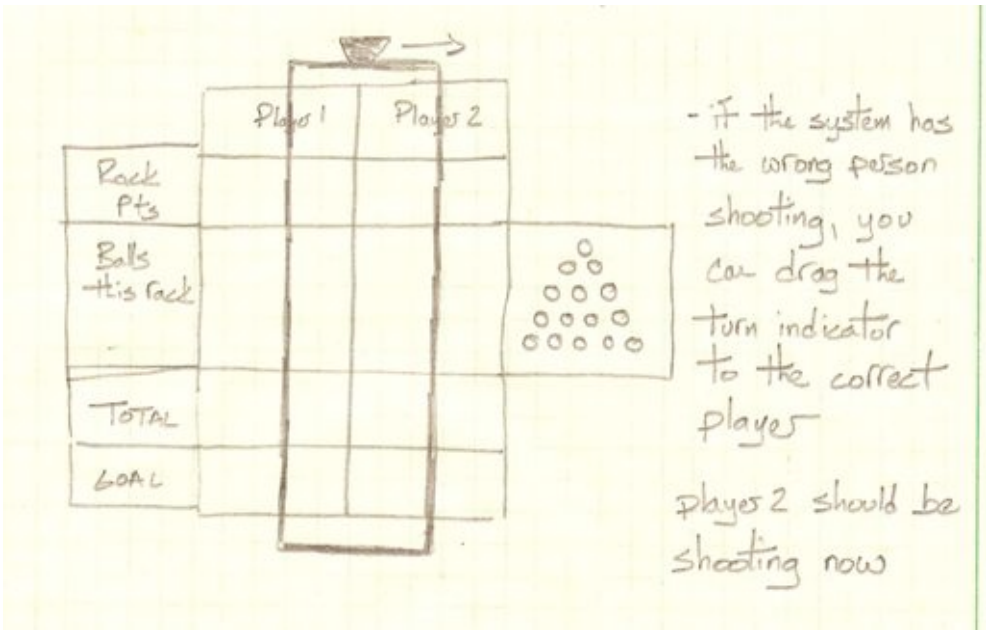

Figure 12: Direct Manipulation to Change Turns

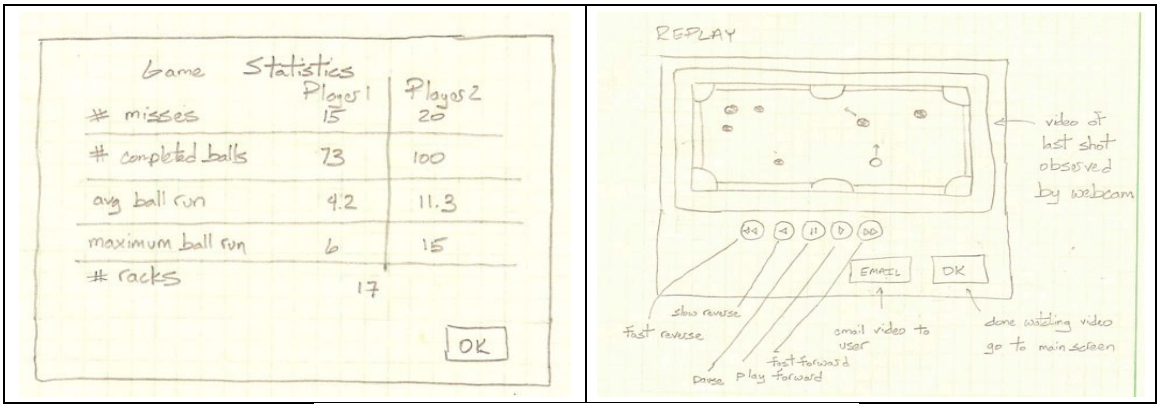

Figure 13: Statistics and Replay Screens

Figure 13 shows the proposed statistics and replay screens reachable from the main interface. The statistics screen simply provides a tabular representation of game statistics that have been collected during play. In this example, basic statistics include the number of misses, completed balls, average number of pocketed balls made per turn, maximum number of balls made on a turn, etc. Gathered statistics can be viewed at any time during or after the game. The replay screen is an exciting feature that will provide a replay of the previous shot recorded by the webcam. Often, during match play, the ability to replay a shot would be beneficial for settling any disputes that may arise during play. Additionally, the ability to save or email replay videos would be a "killer app" feature that would entice many players to use the system. For instance, if a player has made an unbelievably difficult shot during a match, it would be desirable to save that video to show others for "bragging rights." Although players may not choose to use this feature very often, the ability to save and transfer videos of so-called "one in a million" shots would be available when the situation arises.

This proposed design may be implemented with either a touchscreen or a traditional mouse/keyboard interface. A touchscreen would be preferable given the familiarity of many users with touchscreen games that are commonly placed within pool halls, but may be cost prohibitive for a system prototype. Although this design has a fairly sophisticated user interface, we believe that the virtual rack provides a natural mapping between pocketed balls and earned points. This design alternative also allows for seamless integration with any pool table, where no table modification is required for installation or use. By showing each individual ball that has been credited to each player, the system state is clearly visible for the users, hopefully providing low gulfs of evaluation and execution. Unfortunately, the direct manipulation paradigm involves more complex drag-n-drop programming operations than traditional GUI designs.

Conclusion

All of these above designs can benefit from a camera resource although the first instead relies on ball sensors. The potential that ball sensors can be triggered accidentally could be remedied by switching instead to camera detection. All interface styles also have the presence of a numerical score indicator to reveal the precise score at any moment. And all of the systems proposed should require less scoring labor than the average expected in manual scoring conditions. This is made possible by having reasonably low errors rates in game event detection.

Although we hope the user will never have to correct the automatic scoring system, in such a case we want the correction action to be as direct and tangibly satisfying as possible. As such, all three system have a hands-on approach to correcting the score: rolling the score digits, sliding virtual beads, or moving virtual balls. None of the systems have multitasking capabilities so that the users are guided linearly into beginning and exiting pool games. This limits distraction and requires little learning to begin using the system fully. But also, none of the systems handle the ability to call pockets and score accordingly.

The first design differs from the other two in that it does not rely on a screen interface but instead uses mechanical parts for display and score correction. This may be less imposing on technologically-hesitant users as well as the consumption of less power while running and idling. The second system is so very straightforward that it likely cannot be misused. This has the advantage of placing every single interface element on the same screen, allowing maximal visibility of the system at all times. But this simplicity is also the result of sacrificing more detailed information provided in the third design such as which balls belonged to which players. We could also experiment with a form of gesturing to allow users to switch turns; potential gestures include tapping the table or waving the cue stick.

We are currently aiming for the third design in our future work as we see it has the most usability features to explore and will be more fitting for serious pool players in need of such a system.