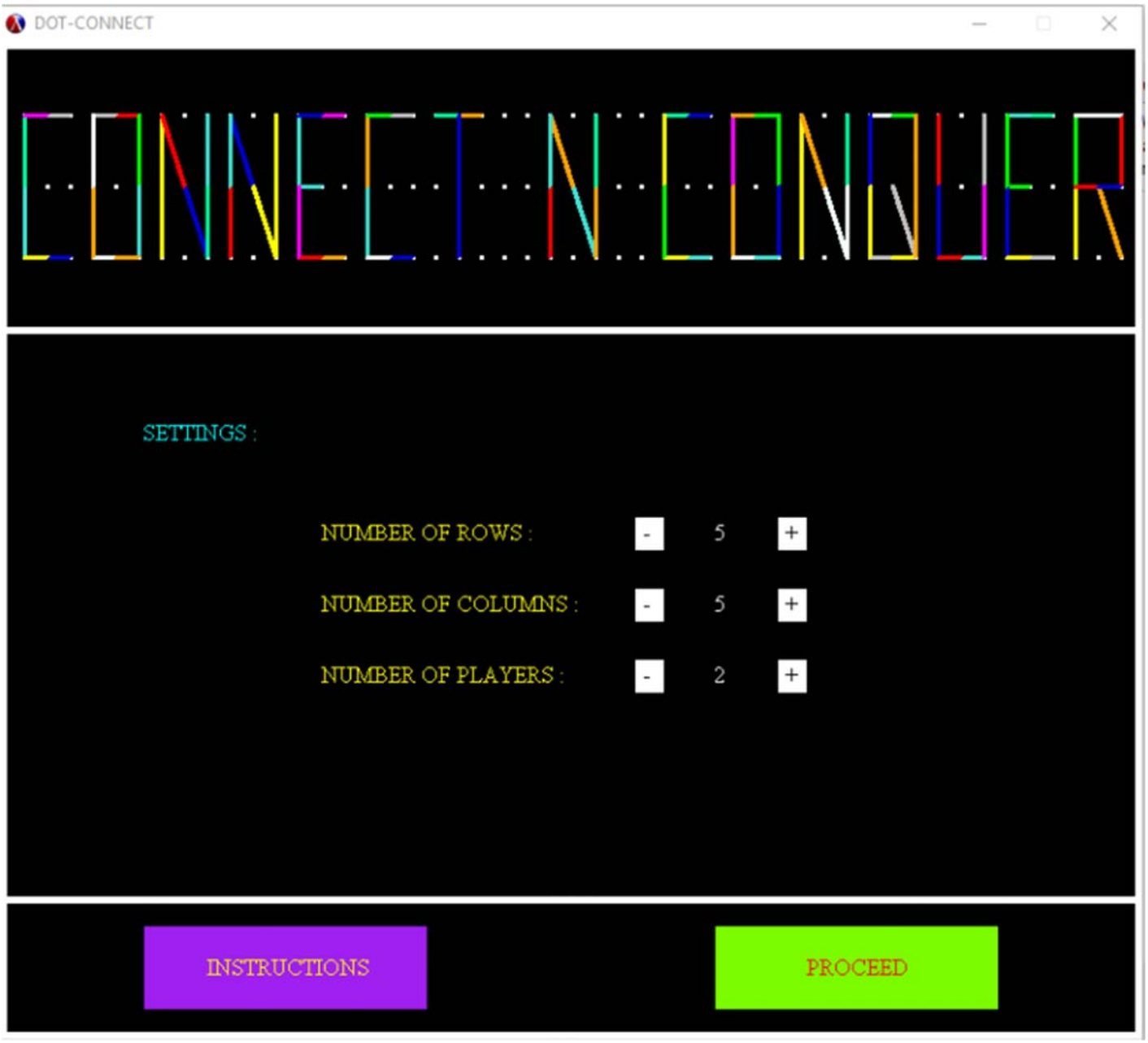


Connect and Conquer

A CS-152 Course Project by-

- *Gurparkash Singh (160050112)*
- *Rajat Rathi (160050015)*
- *Atharv Nandapurkar (160050004)*

- ## Brief Description :

The basic idea is to implement the classic game, popularly known by the names, "Dots and Boxes", or "Connect the Dots", or probably by a dozen other fancy names around the globe. We add another cool name to the list and are calling it, "Connect and Conquer". A catchy name increases the value of anything to about 124.6%, doesn't it?

Anyway, coming back to our game, the game is really simple (or at least, seemingly so). There'll be a grid of dots laid out in front of the players. In every move, a player connects two adjacent dots (only horizontally or vertically). If the connecting of the two dots leads to a square being completed, the player who played that move wins that box and gets a chance to play another move. Otherwise, it's the next player's turn. When all the dots have been connected, the game ends with the player owning the most number of boxes being the winner. Our main aim, while starting with the project was to make the game multi-player, which has not been done before. We searched the Android Playstore and couldn't find a multi-player version.

- ## Delving into Details :

This is the part in which I'm going to talk about how our game gained the form in which it lives today. I will try to cover all the major checkpoints that came en route the final destination. Also, I will cover the roadblocks that were faced, some of which were tackled, while some were ignored (more on that later, in the Limitations Section). So, let's get started!

We used the "Legacy Library" to implement the Graphics, which although, is a subset of the bigger and richer "Racket/GUI Library", but served us rather quite well. Literally speaking, there are only four basic things that you can do using the Legacy Library:

1. *Draw Stuff (includes drawing lines, pentagons and writing Strings).*
2. *Mouse Operations (know where the pointer is, and know where the user clicked).*
3. *Keyboard Operations (know what key was pressed, but only one at a time).*
4. *Use Pictures (function is obvious, but we didn't use this. So there's no point going into details).*

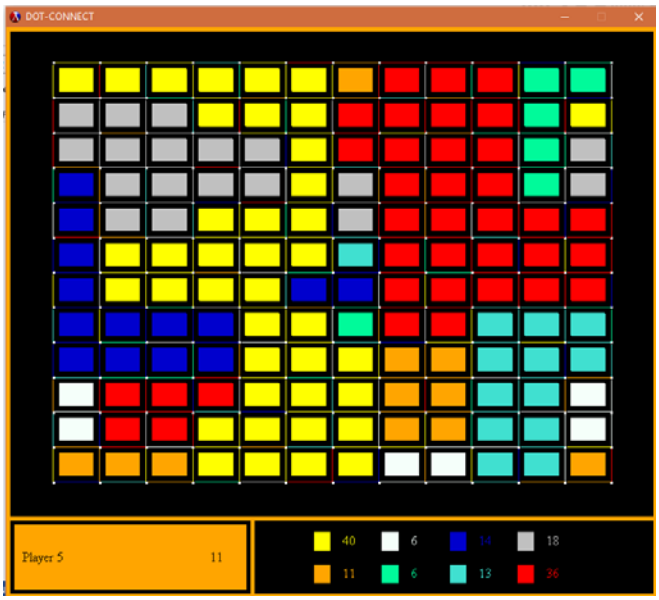
Now, we implemented all the Graphics (including Buttons, Textboxes, Tables), using only the above mentioned three features. I'll get back to how we did it in the "Clever Coding and Designs Section".

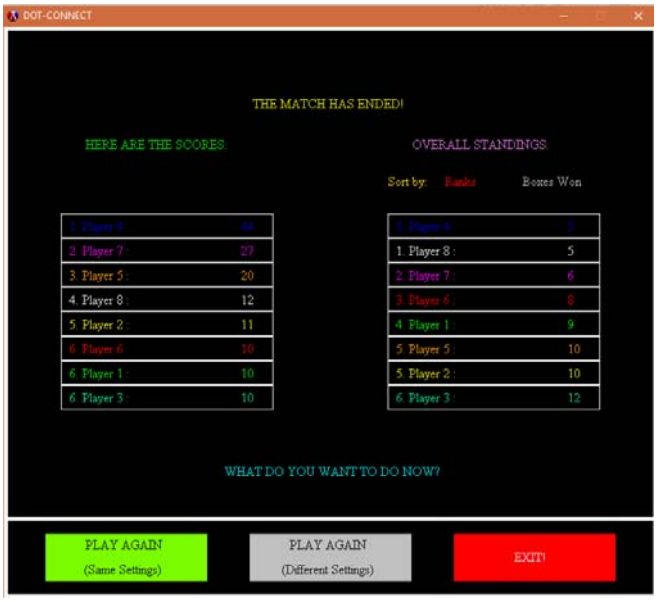
Moving ahead, let's talk about the actual game.

As was planned, we were successful in making the game multi-player. But, not only that, we went a step ahead and made a provision to customize the players. Now, this makes things really interesting as it means that we can have a game in which n number of humans and m number of CPUs compete. Just imagine how fun the game will be when n is zero. All the CPUs competing against each other! Also it is possible to change the intelligence level of the CPU players, on a scale of 1 to 3 (more on that later in the "Algorithms Section"). Because we wanted our game to be flexible, we have kept the size of the grid flexible.

After the size of the grid has been set, and the details of the players have been put in, we are ready to get into the battle-ground.

I'll cover what happens in the battle-ground in the "Algorithms Section". So, let's just move to what happens when a game is over. Here is a screenshot of that.





As it is clear from the above picture, the screen shows the ranks of the players along with number of boxes conquered, on the left side. On the right side, the screen shows the "History". It is the account of all the games played with the same settings. You can sort the overall ranks of players either by total number of boxes conquered or by sum of their ranks in all the games so far. At this screen, you can either chose to play with same settings, which will mean that the performance history will be updated and a new game with same players will start, or you can chose to play again with different settings, which will mean that the previous history is flushed and you can now add more players to the game, or remove some players or modify the players. The last option of Exit exists in case you are tired of playing the game for hours on stretch, or you just want to check if the button really works (it does!).

- ## Algorithms :
  - ### Human's Turn :

    The Human player clicks between the two dots that he wants to connect. Our algorithm finds out the box and the side which the player wanted, and joins it.
  - ### CPU-1's (Easy) Turn :

    The CPU, first checks if there is a box with three sides completed, so that he could make the fourth side and win the box. After that, he will try "Power" number of times to find a move that will not lead to a box becoming a three-sided box. For CPU-1, we have set the power to (sqrt (m n)), where m and n are the number of rows and columns of the grid.*
  - ### CPU-2's (Medium) Turn :

    Similar to CPU-1, it will first check if it's possible to win a box. If not, it will play the move such that the next player gets the minimum number of boxes.
  - ### CPU-3's (Hard) Turn :

    In CPU-3's move, we have implemented, which we call, the Method of Sacrifices. This CPU does not naively take all the boxes that he can win, but also, at suitable positions, sacrifices a small number of boxes, in order to win a bigger chain in the next move.
  - ### Deciding the Best Move :

    To decide all the above mentioned moves, we create a temporary "Invisible Player" and let it play various moves. We see what the best move was, and then make the CPU play

that move. The color of the invisible player is the same as that of the background, so whatever he plays is not visible to the players.

- ## Clever Coding and Designs :
  - ### Buttons :
    As I mentioned earlier, there is no inbuilt provision for buttons in the Library. So what we did was, that we made rectangles of desired colors and wrote on them what we needed. Because we know their exact positions, and we can find out the position of mouse click, we can perform the desired action. Also, we have added an effect on buttons whenever the pointer is above them, to make them look more realistic.
  - ### Textboxes and Fields :
    Every text input follows a standard position. Take a key press, add the key's value to a string variable and display the string on the required position. If that key was a backspace, remove the last letter of the string (if it exists), draw a background colored rectangle on the screen, and write the updated text on it. Recursively call the function again, or terminate if the key pressed was the Enter Key. We have added a cool effect on textboxes too. Implementation of fields was on a similar note too.
  - ### Multipurpose Randomize Function :
    We have made a function that randomly sorts any vector. We have used it in two very different situations. One is to allot the players their turns. Another is to randomly allot colors and decide the sequence in which the title at the home-page is drawn. So, the title at the home-page will look different every time you play the game.
  - ### Deciding the Box and Side in Human's Turn :
    We have referred to every point in the grid, not by its exact co-ordinates (x, y), but by (n, m), where n and m are both non-negative integers. This seemingly simple modification, saves us a lot of computational time. Because, now to find the closest dot to the point clicked by the user, we don't have to check the distance between that point and all the other points. We can simply use the greatest integer function to find n and m, and from that we can find out x and y. Once we have the closest box, we can find whether the user wanted a horizontal or vertical line from the slop of the line passing through the dot and the user-clicked point.

- ## Limitations :
  1. *The method of Sacrifice, works on the principal that a player sacrificing in the current move, will get a large number of boxes in the next move. Now, as the grid becomes smaller or the number of players increases, the player making the sacrifice might not even get another turn. So, CPU-3, won't win comprehensively if the conditions don't meet.*
  2. *Also, we weren't able to make "the-ultimate-CPU", i.e., players proficient in the game may be able to defeat our CPU-3.*
  3. *The Text-Box implementation isn't perfect, because of how the Library works.*
  4. *Sometimes (very rarely), the game freezes, for reasons seemingly unknown.*