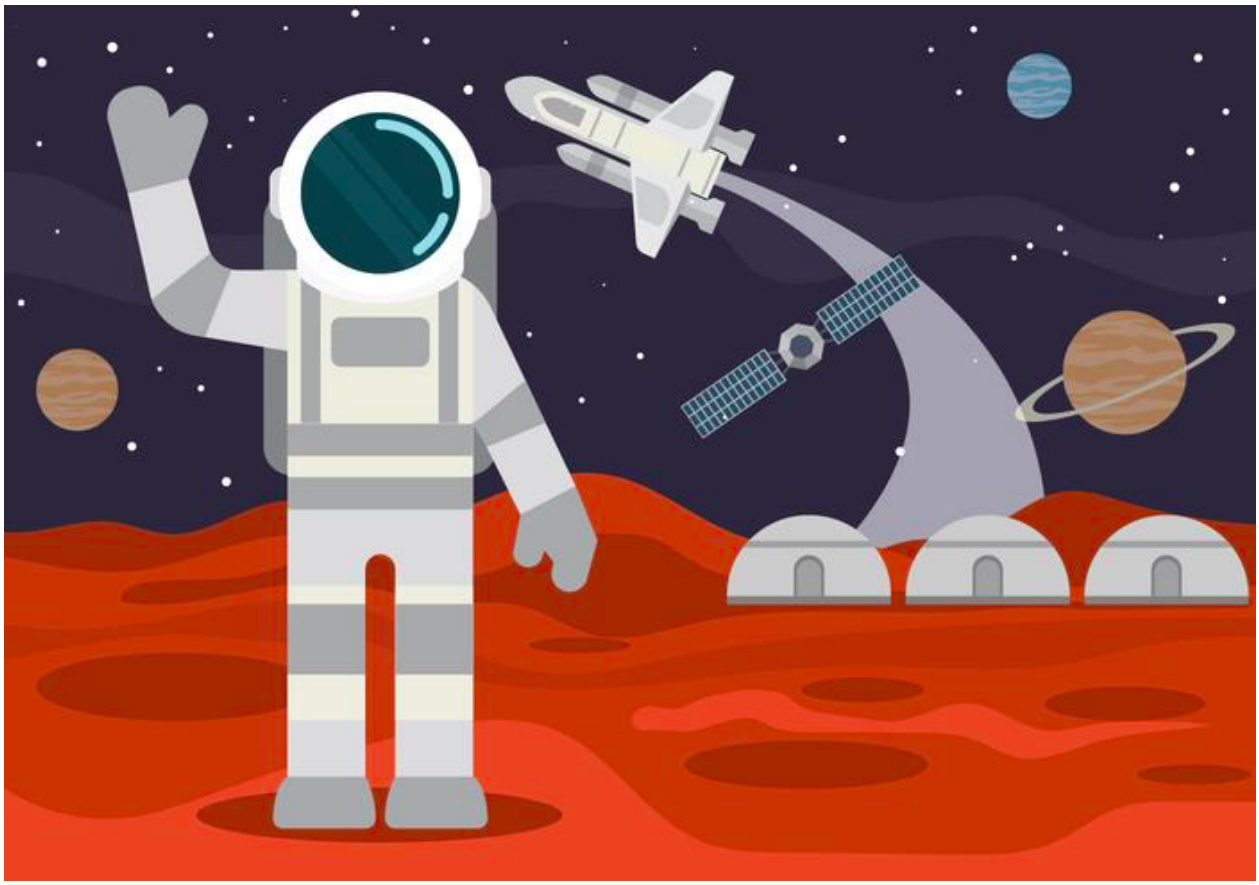
Project Report

Entertain The Crew



Team: PERSEVERANCE-20

This is a project report or a technical document that is going to help you understand the project better. It has all information in a very organized and detailed manner explaining the entire workflow of the project.

We hope this helps you!

TEAM: PERSEVERANCE-20

TEAM MEMBERS:

1) OSHIN SAINI

Github: https://github.com/oshinsaini

LinkedIn: https://www.linkedin.com/in/oshin-saini-838718184

Mentor Assigned: Ms. Durgashini Nadarajah

2) SIDDHI MISHRA

Github: https://github.com/Sidsag

Linkedin: https://www.linkedin.com/in/siddhi-mishra-42479717b

Mentor Assigned: Mr. Ketan Arora

PERSEVERANCE MEANING

the ability to keep doing something in spite of obstacles. We wanted to do justice by our team's name and here we are :)

As the program revolved around MARS.

(THE MARS COLONIZATION PROGRAM) we did some research and this 30th July, Perseverance rover is going on mars to do wonders. We also wanted to do wonders with our project :)

Table Of Content

Overview

Project Title: Entertain the Crew

Project Name: Tic-Tac-Toe(new version of traditional tic tac toe)

This project is made under the Microsoft Mars colonization program organized by Microsoft.
It is a game website that provides tic tac Toe games with additional features. This project is made under consideration of providing an entertainment platform to our crew members during our Mars colonization program. We have designed different versions of the game using terminology level 1/level 2/ unbeatable etc. There are 6 different versions of the same game we are providing on our website. The main highlight of this project is we have developed a new class of a game using the same set of rules of tic-tac-toe but have a 3D version with some additional features and rules.

Approach:
We are using a functional programming paradigm in our project. Our agent function & all other functions are either called by the front-end or any function of the backend to perform the desired task. these function providing functionalities like changing the HTML table values /making computer move/performing the changes on click/checking game condition etc.these functions providing support as the back-end of our project.some of these function are designed using some basic algorithm to perform a particular task or some using AI algorithms for achieving their agent function goal. Detailed descriptions of all the functions used in our program module are given in our next section(refer Functionality page for the same).

Goals

Task: In this project, our task is to design a game website that entertains the crew members during their mass settlement program. For this purpose we have to provide traditional tic-tac-toe game with the following modification:
1) make a program which makes possible to computer to move instead of one player
Requirement: it might be possible that during the Mars colonization program the crew members may not have a partner at the same time they want to play the game so it will eliminate the dependency of another player in a traditional tic-tac-toe game.

2) use AI algorithms as an assertion to make our computer move in a smarter way.
Requirement: this will improve the level of the game and also help crew members to enjoy it as a challenging game.

Challenge: In this task, our team takes a challenge to take the project to another level means convert the traditional tic-tac-toe game to a different game that follows the same rule as traditional tic-tac-toe along with some additional rules and functionality which increase the difficulty level of the game.
Requirement: our team thought that it will be good if we can change the game with the provided one, will it enhance our creative & technical skills along with that we can provide a different game experience to our crew members which may help them to think more analytically during the game.

Therefore from the above discussion, we categorize our project goal as below:
1) design a simple tic tac Toe game
2) modify the game by making a computer as a one-player game.
3)Smarten the computer moves using AI agents to different levels of the game till unbeatable one.
4)modify the current tic-tac-toe and design a next-level version of the traditional tic tac toe game.
5) make the computer as one player in our design game.
6) smart and the computer move using an AI agent to different levels, which take our modified rules into consideration.

Requirement Analysis

Taken our project goal into consideration our team have finalized the following requirement for our project:
1) framework of the website
2)designing an algorithm for each version of the game and integrate it with one base website
3) backend language to implement our algorithm.
4) front end elements to design our website.
5) a platform to work on the project in collaboration.
6) a platform to deploy our website
7) Design a website that should work on different browsers like chrome /Microsoft edge.
8) Provide user convenience, that is the user can go to any level of the game at any time and can restart the game, etc.

our team did analysis and research on resources for satisfying the above requirement we design following technical specs for our website:

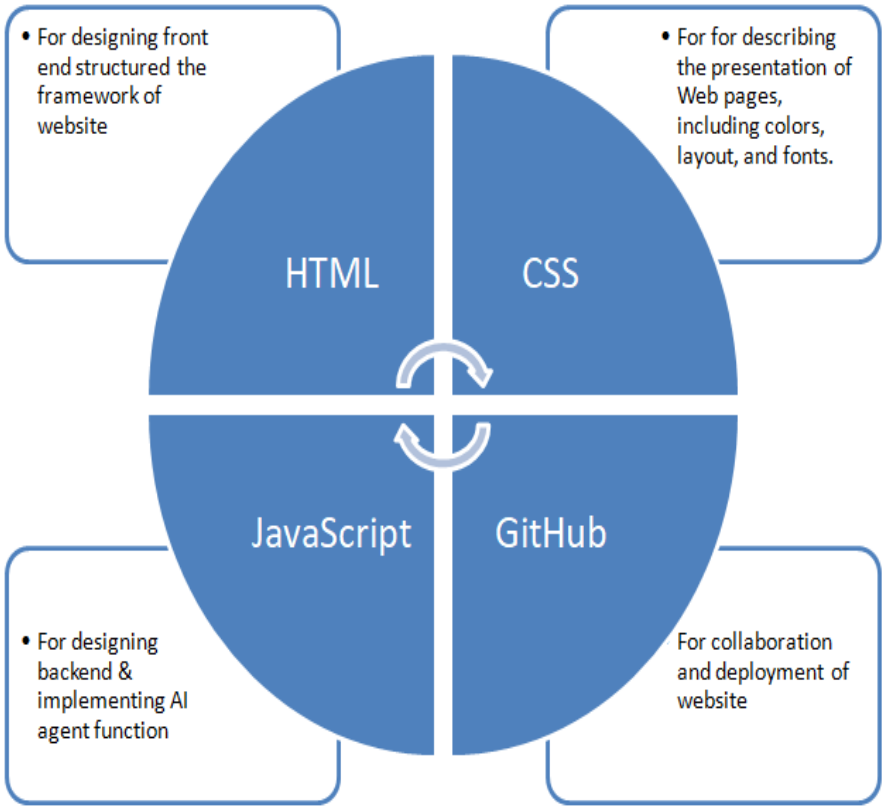


Fig R.1 Technical Stack of Website

our team has decided to work on each and every version of came in phase-wise and for each version the workflow will be like following:

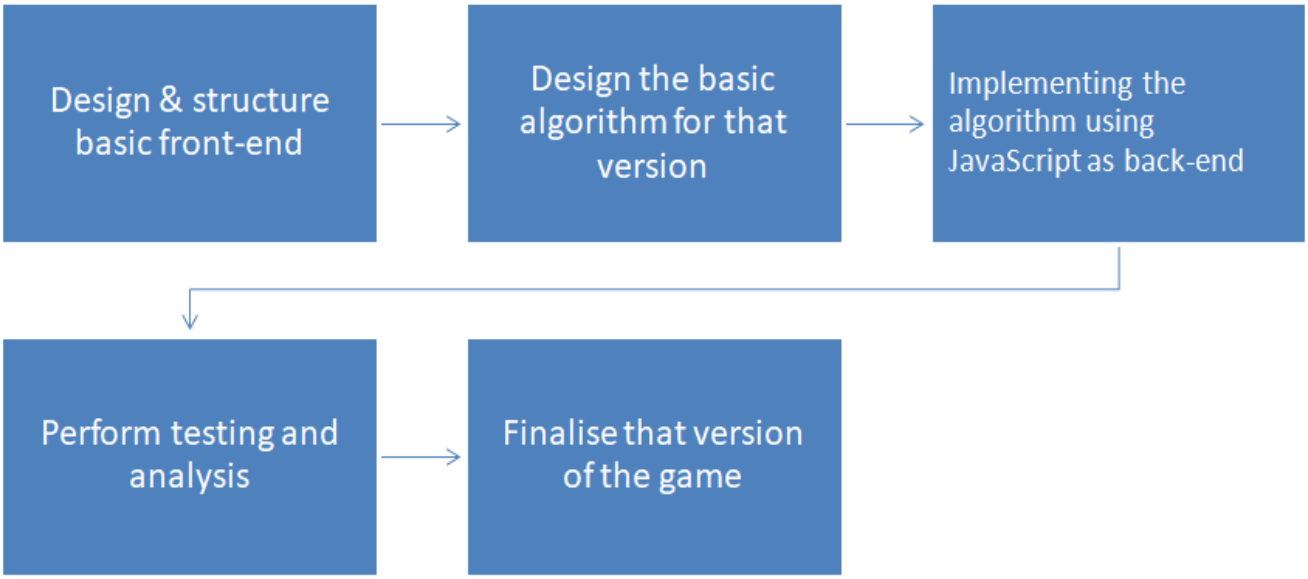


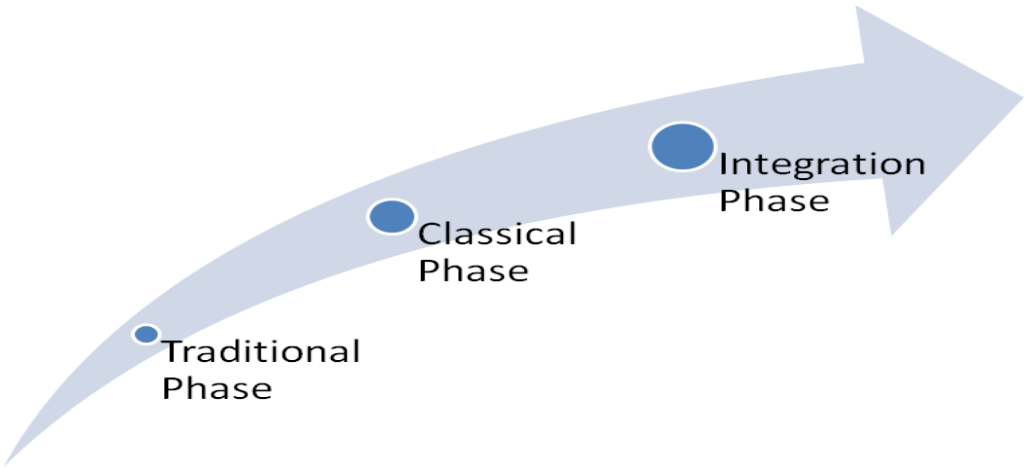
Fig R.2 workflow for each version of the game

We are using the same workflow for all the versions of a game and in the end, we will integrate all the versions on a website and deploy it using Github**.**

Phases

As discussed in our requirement specification we have divided our project into phases in order to do work on different parts of our project in a sequential manner.

We have decided to divide our project into three different phases:
1) traditional phase
2) classical phase
3) integration phase



P.1 Phases of the project

These phases have their own subphases as design and development phases.
For a different version of our project we categorize them into two phases:

1) Traditional phase

In the first phase we have designed and develop traditional tic tac toe into a different version which is the following:
1) with two-player mode.
2) level 1 of the game where the computer is able to make moves instead of one player.
3)level 2 in which the computer trained using some agent function to make smart moves.
4) In the last version of the game, we are using the minimax algorithm which will convert and implement according to the functional requirement of our base game to make the computer move unbeatable.

2) Classical Phase:

In the second phase, firstly we design and build a game that follows a set of rules of traditional tic-tac-toe game but has a function and some change in the rule which takes the game into the next level.
1)The first task is to design and develop a 2 player version of the game and verify whether it follows all sets of rules of the traditional tic tac Toe game.
2)now analyze the two-player mode and change this version to make a computer move possible.
3) Smarten the computer move using some basic AI function to develop the two versions of the game version called level 1 and level 2 respectively.
4)for an unbeatable version, by taking a reference of the traditional minimax algorithm and developing an unbeatable version of our designed game.

3)Integration Phase

After the traditional and classical phases of development, we will go to our integration phase.
1) Here we structure and design our base page where all other levels/versions of a game can be called. The aim to design the base page is to integrate all the versions of the game on a single platform.
2) do work on lookup and beautification of websites.
3) deploy the website using the GitHub platform.
4) test our website using different browsers and test cases.

Content

Following is the content of our website:

1)Base Page

2)Play with Friend (2D)

3)Level 1 (2D)

4)Level 2 (2D)

5)Beat me if you can (unbeatable version)

6)Understanding the game better (3D)

7)Play with Friend (3D)

8)Smart Level 1(3D)

9)Smart Level 2(3D)

10)Unlimited

The above content is actually a different version of a game discussed in the document earlier. For every content of the website, we have designed different levels/forms of the same game so that users can experience different flavors of the same game. every level is totally different from the other level of a game and is only dependent on other levels in terms of their rules and some features. The motivation behind designing a 2D version of a game instead of having the next level of a game to provide users with a set of variety and choices. every level has their own code and design which are integrated to the base page so that they can go back to the base page at any time during the game using “back to the base button”.we are also providing the facility of "restart game button" that helps the user to leave the game in between and start a new game.

Note: detailed functional specification i.e. User guide is attached with the document.

Functionality

i) Made a game algorithm which allows the user to play 2d Tic-tac-toe, in two-player mode using basic algorithms.

ii) Design the algorithm to make a computer to move in place of one player. We have done it in three Versions:

1)In the first version, we have used simple logic of move using JavaScript as backend & HTML/CSS in its frontend (i.e. making random moves) for our computer as level 1 of the game.

2)In the second version, we have designed a CSS based computer move which makes our algorithm unique and different & difficult from level 1.

3) In the third version, we have used the popular “minimax algorithm” to implement our level 3 of the game. We have made some functions to get board value and change the index using the id of the table; here we use JavaScript for backend development & HTML/CSS for frontend.

iii) Design the next level of the game, we have taken reference from level 3 of tic-tac-toe to design the six faces using a table and providing these table id to the backend to render the changes during the game. We have made a board as a variable to define all the faces of cubes and do change in it with help of bS array which enables the board to follow the rule of the game during the change in values of a plane.

1) In the next step, we design an algorithm to make the computer move just seeing win/lose condition and move sequentially and implement using JavaScript. Here our computer move is made by calling the best move function.

2) Increasing the level of difficulty we made our computer to move in a random manner using random ()/random. Fixed () etc. and made level 2 of the project.

3) In our final version of our project, we have made an algorithm that makes our computers make smart moves in our designed game. Here we take analysis and then make reference to the “minimax algorithm” we have developed our final AI algorithm & implement using JavaScript. For the completion of this phase, we have to undergo multiple testing & debugging phases to make our algorithm unbeatable.

overall functionality is discussed along with code in the next section.

Traditional Tic-Tac-Toe(basic functionality:)

Following is an explanation of the two-player mode game function, this is the framework that we are using in all other versions of the game with required modification.

startGame(): this function is performed following task:
i) initialize all the value of the HTML table for the game board;
ii)decide the turn.

setMessage(msg): used to print the game message on the screen

nextMove(square): firstly checking the winner, if it's not true then it checks for the index clicked is available on not, if available (make the turn), otherwise, give the command to choose another board.

switchTurn(): it is calling following function:
checkforWinner() //if it is,print the winning message
Otherwise check for tie using CheckforTie(), if both conditions are not satisfying it will switch the turn between X & O.here messages are printing use**setMessage(msg) function .**

Some other useful function of this programme is following:

checkforWinner();	**//**check for winner by calling function checkRow
CheckforTie();	//check for tie
checkRow(a, b, c, move);	**//**Call to check if the row is filled
getBox(number);	**//**Return box contents
clearBox(number);	**//**clear contents in box

Level 1(traditional tic-tac-toe:)

We have used following function in back-end:

//to take player's move
addPlayerMove():it' s firstly checking the board's available or not(by checking win condition & blank space);
It's calling following function in order to render the changes and check winning condition:
render_board(); //render changes on board
check_board_complete(); //check weather board is complete for tie
check_for_winner(); //check for winner

After that it is calling the computer move function, to make the computer move as next turn.

//to take computer's move
addComputerMove(): this function randomly selects any index using while loop till an available index found in board after that it reder the changes and check for winner using following function.
render_board(); //render changes on board
check_for_winner(); //check for winner

//render board
render_board (): it first deploys the changes into the Html table and also changes the value of our local board(play_board).

Apart from above we have used two button function:
reset_board(): reset the HTML table value as well as our local board's index values.
restet_match(): it is calling reset_match() to reset the board/table and also changes player & computer score to zero.

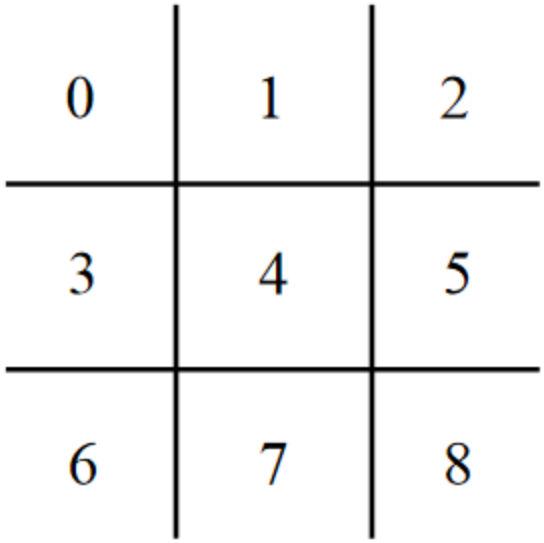
As assistant of check_for_winner we have used following function :
check_for_winner () ---------->check_match() //to check if game won or not------>check_line() //to check for particular line in game won

Level 2(traditional tic-tac-toe:)

This is using HTML and CSS to perform the functionality of the game. We have added this to our project to remove the repetition of the same functionality at different levels and add some new flavor to our project.
Special Feature: We have only used HTML, CSS for making the tic-tac-toe work. We did not use any JavaScript in any of the functions. The CSS smartly works and tries to win the game.

The computer checks for three major conditions:

1) WIN CONDITION:
   The computer always checks for a condition where it could win in any case.
2) LOSE CONDITION:
   The computer will try to save its move by defending any win. It will block every winning move of the opponent.
3) SPECIAL CASES:
   It will check for some special conditions defined by us. If we number all the cells on the board.



   The computer will check for diagonal matches
   a)If 0,4,8 are selected the computer will mark on either 2,6 to either go for a winning Move or tie move to set a trap.
   b)Similarly, If 2,4,6 are selected the computer will mark on either 0,8 to either go for a winning Move or tie move to set a trap.

4) RANDOM TURN:
   Lastly, after checking all conditions the computer will make its turn and place a safe random spot.

Level 3(Traditional tic-tac-toe:)

At this level, we are using a minimax algorithm to make computers move unbeatable. To get these functionalities we have modified our game function as follow:
We are using a reference board(bS array) whose value will change by click function and it will refresh the board (ref board) & render the changes in the HTML table. All the details for every function & their functional flow are given below.

startGame(): this is a function to start the game by initializing each value of the bS array and using these key values, initializing the Board indexes. In the end, it is checking the current player and calling the move() function.

move(): it is a function to call the players depending on our global variable and it is also used to switch the computer move when it is called by click function().
It is calling following function:
BestMove() //when current player is computer I.e. “0”.
SetTimeout(built-in) //to provide time lag which is used by the computer to make its move.

Click function(built-in): it is used as a response function for click made by the user on the board.it does following functionalities:
1.check current player is “X”(i.e. human) and is the board available at that place, if it is, then it will called to render the changes using following function:

```
refreshBoard();      //to make the changes using the bS keys on board
move();              //to switch the move
playJumpSound();     //to indicate the jump
render();            //render the changes & check win/loss/tie etc.
```

Explanation of each of the function is as follow:

refreshBoard(): it refreshes the board values with new key values of bS board and calls renderBoard() function.

renderBoard() : it renders the changes made on the refboard to the HTML table using cell_id.

render() : this function is calling three different functions to check for win/tie condition by checking won variable & calling tieA() function and give resultant message accordingly or initiate the turn.this is used for displaying the message on screen about the game current condition.

tieA(): this function is checking the reference board for tie conditions during the game by checking every index of board status.
gameWon(): it is checking the game won by checking all the conditions of winning the game.

bestMove():
This is the function that performs the computer move and uses a minimax algorithm to decide the computer move which makes the winning probability of the computer about one.funtionality of minimax algorithm is discussed in detail in the next section.
Here first computer goto all the possibilities of the board using the base for loop, check if it is available to make a move then it calls minimax function by passing Board & maximizing condition as false & store the result in score variable which update the best score & bsfinder in turn. In the end the final best score and fbsfinder is updated and renders the changes on board in the same way we are doing in the click function.
Note: in every case, we backtrack the value of the board after calling minimax to explore all the possibilities.

minimax():
It is a recursive function which is calling itself alternatively depending upon maximizing condition is true or false. It have following base condition:
1)if isMaximising is not true && gameWon() returns true it will **return +1**
//as O(i.e. computer) have turn in this case of function calling & our goal is to maximize “O”
2)if isMaximising is true && gameWon() returns true it will **return -1**
//as X(i.e. human) have turn in this case of function calling & our goal is to minimize “X”
3)if tieA() returns true this **return 0**,irrespective of turn during the function call.
So using the above we return the possible outcome of after the particular move of “O”.

We are also using the following button & click function:

("#restartButton").click(function ()

playJumpSound()

playErrorSound()

playWinSound()

Classical Tic-Tac-Toe(basic functionality:)

As discussed above to take our game to the next level we have designed a tic tac toe 3D using a Rubik's cube concept here we are using six faces of Rubik's cube as our board and modify some tic tac toe rule, as if a move is done on a corner that it will be deployed on the other side of a cube also and this way we have designed our next level of tic tac Toe using following functionality.
to attend the functionality of render the changes on the corner of a cube also we have design a variable array in which we define key values by calculating the all related available cells is 26, so there are 26 different key values available in **bS array**, keys values rendered the changes on the boards' planes using bS.keyValue_name function.
We have defined planes as a variable and deploy/define their colors accordingly and also we are using translateLetterToColor() and translateColorToLetter() to change the letter of cases to a corresponding color or colors as cases to the corresponding letter respectively.

startGame():
In the start game function firstly we initialize all the values of bS array/ won and current players and also we define globally boards for all six planes which is related to their corresponding HTML table and obtaining their values using key values of bS array.

move():
In the move function we are checking the game one condition and after that, we are switching the current player between X and O.

Click():
In the click function as we have discussed in a level 3 of traditional tic tac toe above we are calling the same set of functions with the same functionality as discussed.

gameWon():
gameWon function is again checking the game-winning condition by taking each plane into consideration using for loop & getBoard() function and checking the all set of game-winning conditions on it and if in any plane the game-winning condition is true it otherwise it returns false.

getBoard():
Get bored is a function which we used as an alternative of a three-dimensional array to get the corresponding to the particular index as cases.

render():
render function is to display the key current game condition on the screen.

refreshBoard():
Refresh the board is used to refresh all the board planes using key values of bS array. It is called after each move and it alternatively calls **rendorBoard()** function to render the corresponding changes into the HTML table.

displayPlanes():
display Plane is a button function that displays the corresponding plane which is being clicked.

Along with the error function we also called following function:

```
//button functions
("#restartButton").click(function ()
("#orangeButton").click(function ()
("#greenButton").click(function ()
("#purpleButton").click(function ()
("#blueButton").click(function ()
("#redButton").click(function ()
("#yellowButton").click(function ()

//sound function
playJumpSound()

playErrorSound()

playWinSound()
```

Classical Tic-Tac-Toe(Smart level 1:)

Note: all the function used in this level are already described in the above section of classical tic-tac-toe description, the only new two fusion we have included in this part to enable computer move as follow:

bestMove():
This function is called by move function when the current player is a computer(i.e. “O”).in this function we are using three modules:
1)check if “O” wins: module is checking all the possible moves available, the possibility of winning the game and if it is, then it will be the final move and changes will render on board.
1)check if “X” wins: module is checking all the possible moves available, the possibility of losing the game and if it is, then it will be the final move and changes will render on board.
3)if the above condition is not satisfied, the computer will take a sequential move as its next move.

Classical Tic-Tac-Toe(Smart level 2:)

Note: all the functionality of this level is the same as above except the changes have been made in the third module of bestMove() function.

bestMove():
Modification: in the third move instead of going sequentially the computer is taking random() moves, the logic behind making the move smarter by the random move is now the user has to take the trace of all sixth planes available in-game which makes this level harder than previous one.

For achieving the random move functionality we are using the following function:

```
Math.round(Math.random()*10)%5;   //to select board plane
Math.round(Math.random()*10)%8;  //to select bS finder index
```

If the move is not available, the computer will take the next available move in a sequential manner.

Classical Tic-Tac-Toe(Unbeatable :)

Note: all the function used in this level are already described in above section of classical tic-tac-toe description, the only new two fusion we have included in this part to enable computer move as follow:

For this label, we are using the same approach of traditional tic-tac-toe, unbeatable version expect that here our task is challenging because every move corresponds to the change on another plane so the computer has to think the possibility of winning or losing the game on every move with respect of all the plane as the number of board is increases and all the board is dependent on each other with respect to move. Therefore we have to convert our present function into another level in which computers by selecting a single level can able to find out the changes in another level and can predict there is a possibility of winning all losing the game using a minimax algorithm. For getting the bold plane during our best move function we have designed a function **getBoard()** and for obtaining the effect of the changes on all board plane we are using another array **refBoard** which have a key to refer the bS key array which ultimately changes the board planes values.

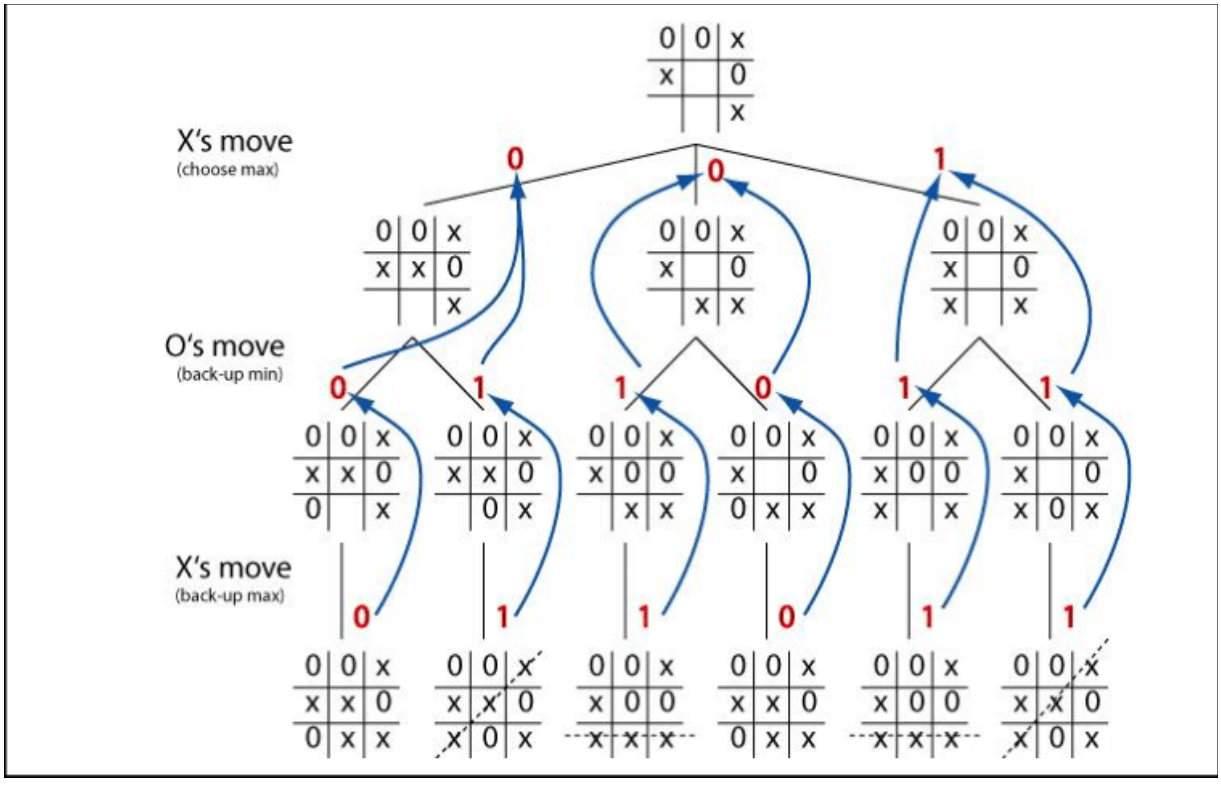
Modification in bestMove():
In our best move function, we have created to two variable final best score and best score similarly best finder and final best finder in this way we are exploring every plain and obtain the best finder and best score for the particular plane and checking the score and updating the score with our final best score and final best finder. further bestMove functionality is the same as traditional tic tac toe ultimate level.

Modification in minimax():
Minimax function is also same, except that in this we are in a gameWon() we are going to check all the plane and similarly for a tie, we are going to check that board by passing it as a functional parameter and for the particular board we are checking the game won or tie for that board and returning the value correspondingly as discussed in the above section.

Minimax algorithm

Minimax algorithm is also same , except that in this we are in a gameWon() we are going to check all the plane and similarly, for a tie, we are going to check that board by passing it as functional parameter and for the particular board, we are checking the game won or tie for that board and returning the value correspondingly as discussed in the above section.

Minimax algorithm is also same,except that in this we are in a gameWon() we are going to check all the the plane and similarly, for a tie we are going to check that board by passing it as a function parameter and for the particular board we are checking the game won or tie for that board and returning the value correspondingly.



M.1 minimax algorithm description

Browser Support:

Our project is supporting following browser:

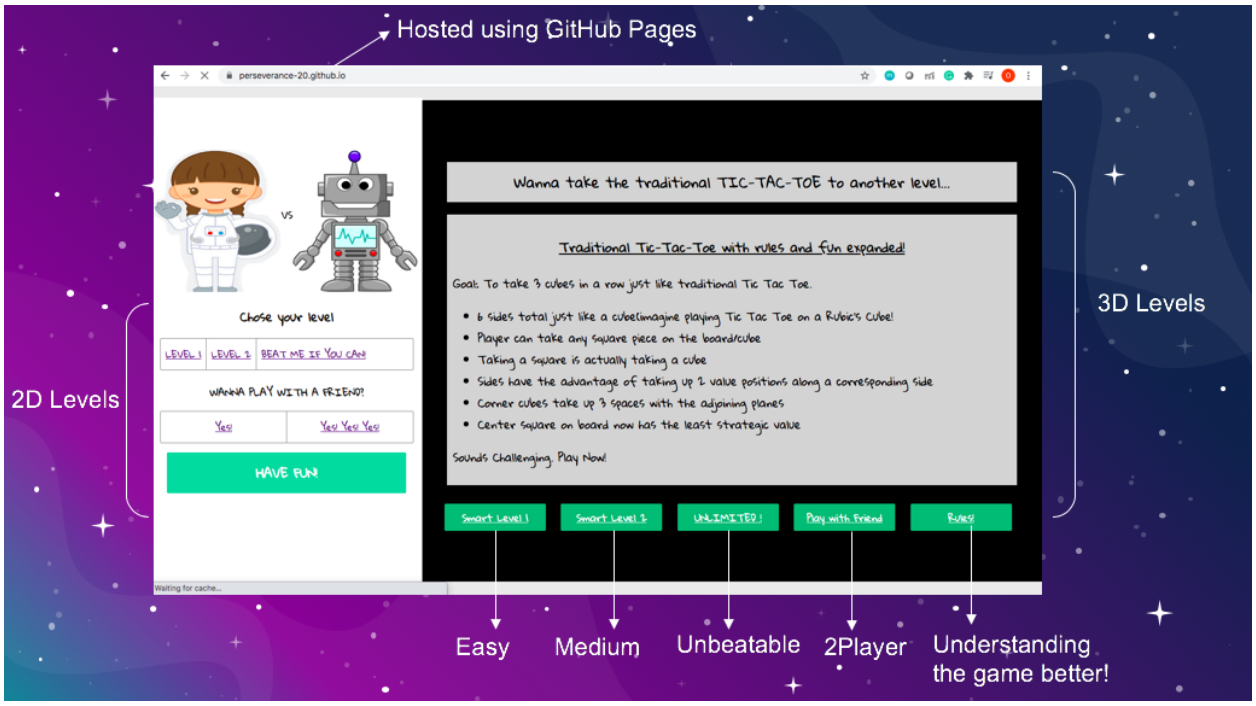
1. **Google chrome**
2. **Firefox**
3. **Safari**
4. **Opera**
5. **Microsoft edge**

For better experience: we refer to Microsoft Edge latest version.

Hosting platform : github pages(https://perseverance-20.github.io/)

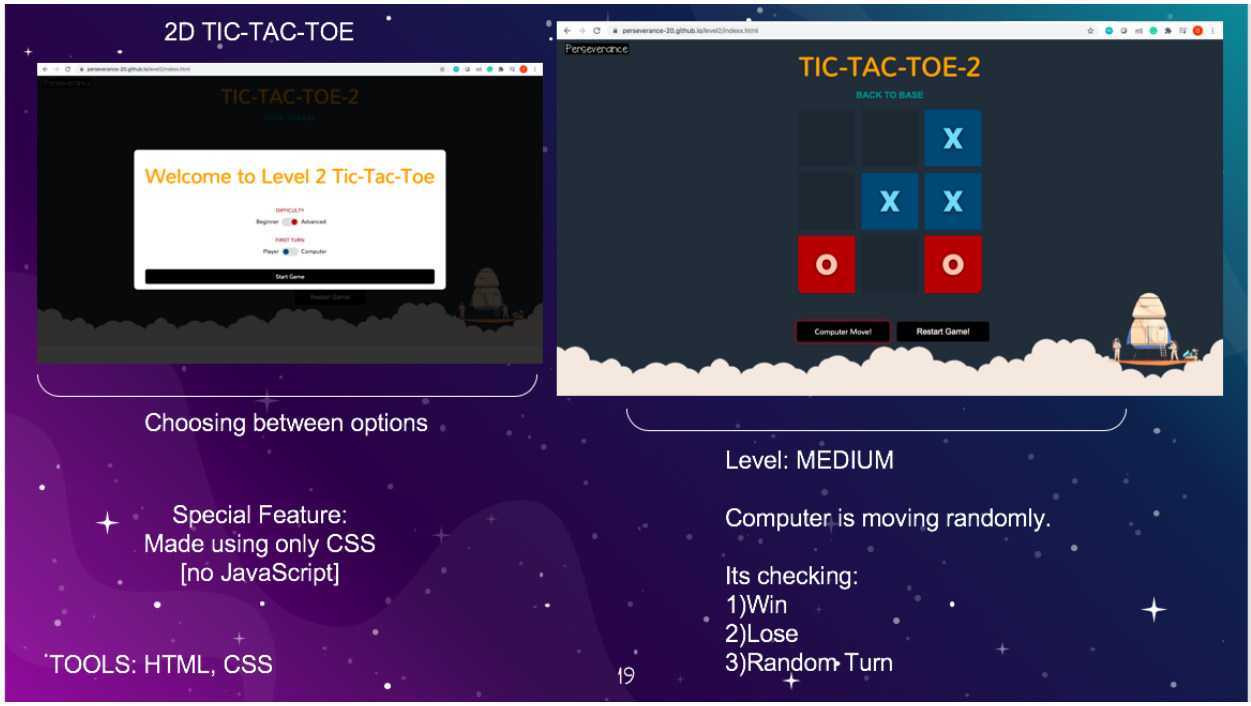
USER GUIDE

HOME PAGE:

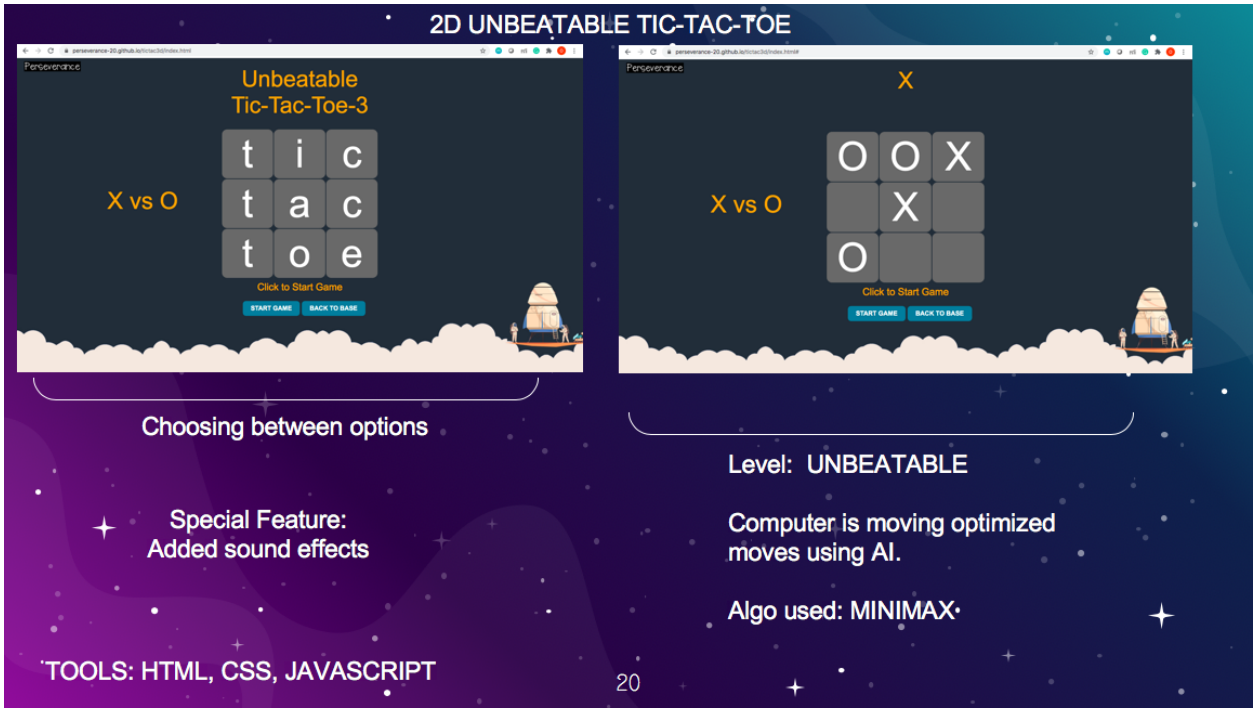


2D LEVEL-1 TIC-TAC-TOE:

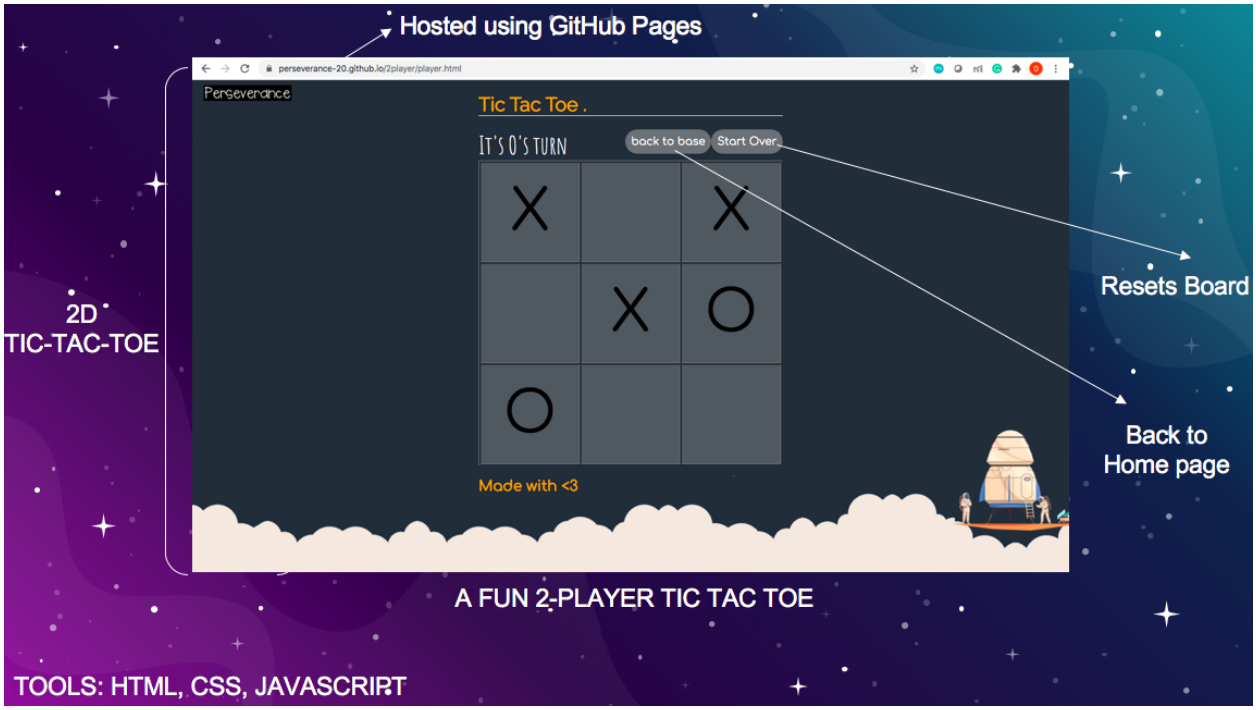
2D LEVEL-2 TIC-TAC-TOE:



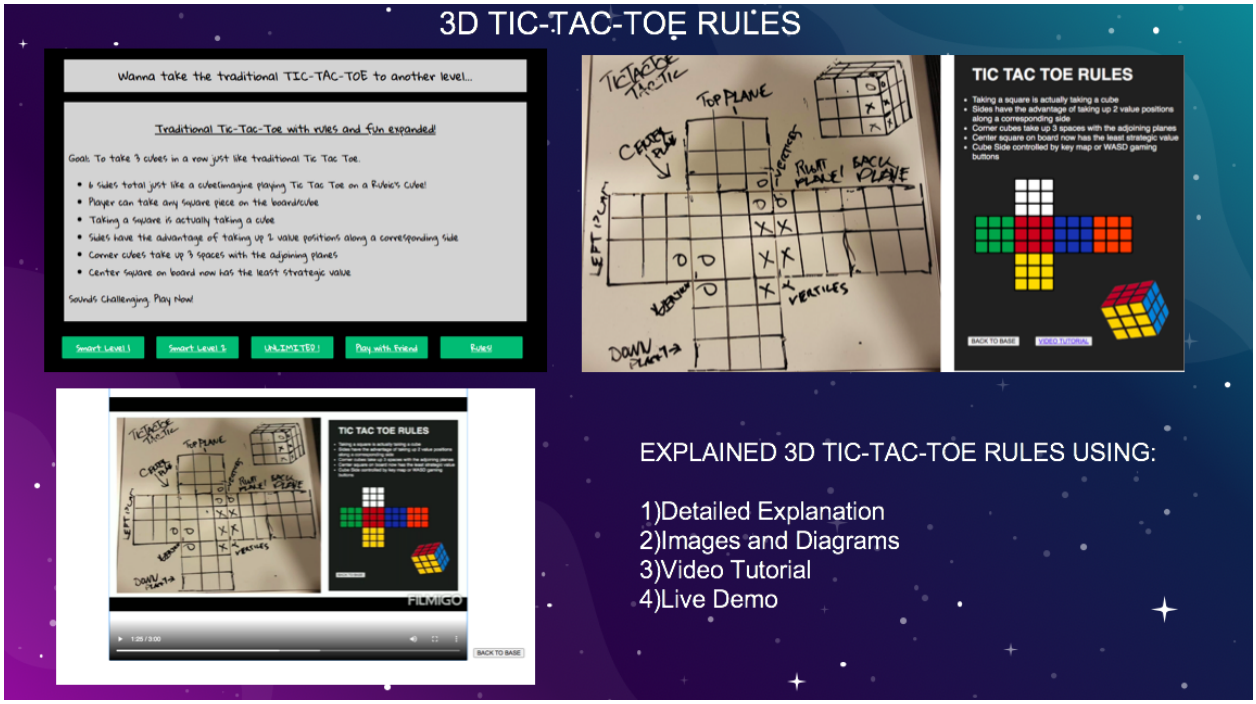
2D UNBEATABLE TIC-TAC-TOE:



2D 2-PLAYER TIC-TAC-TOE:

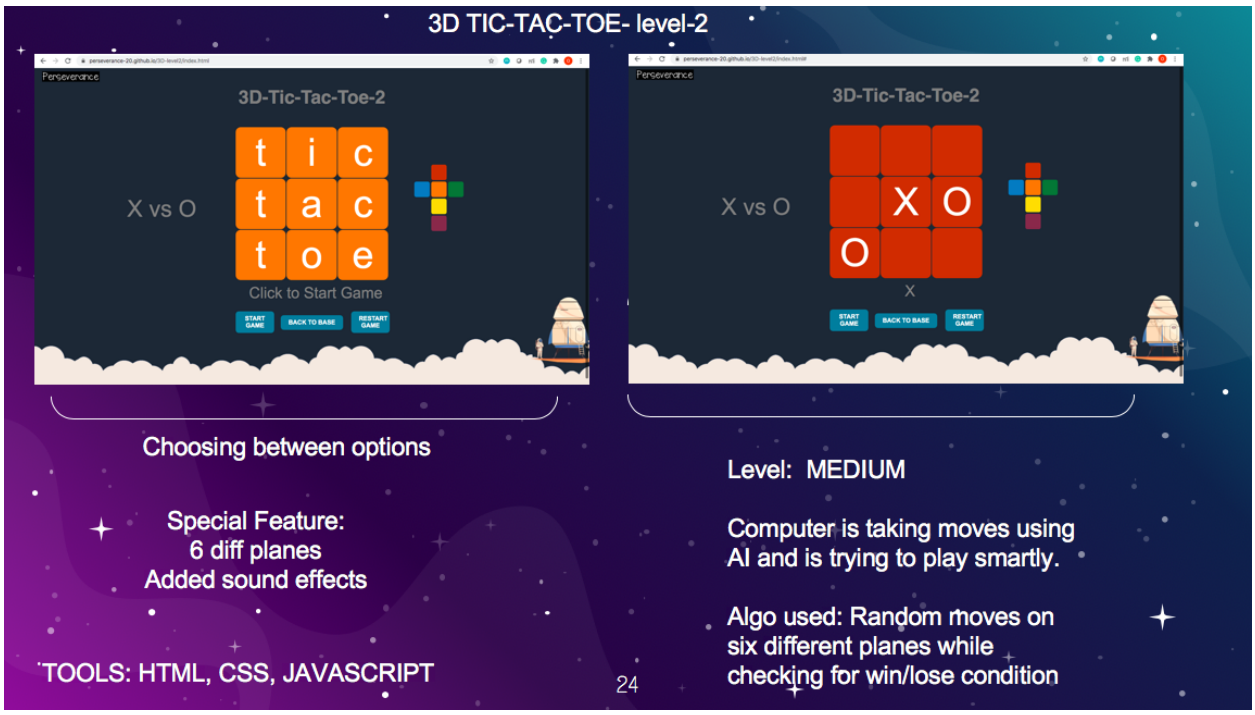


3D LEVEL RULES:



3D LEVEL-1 TIC-TAC-TOE:

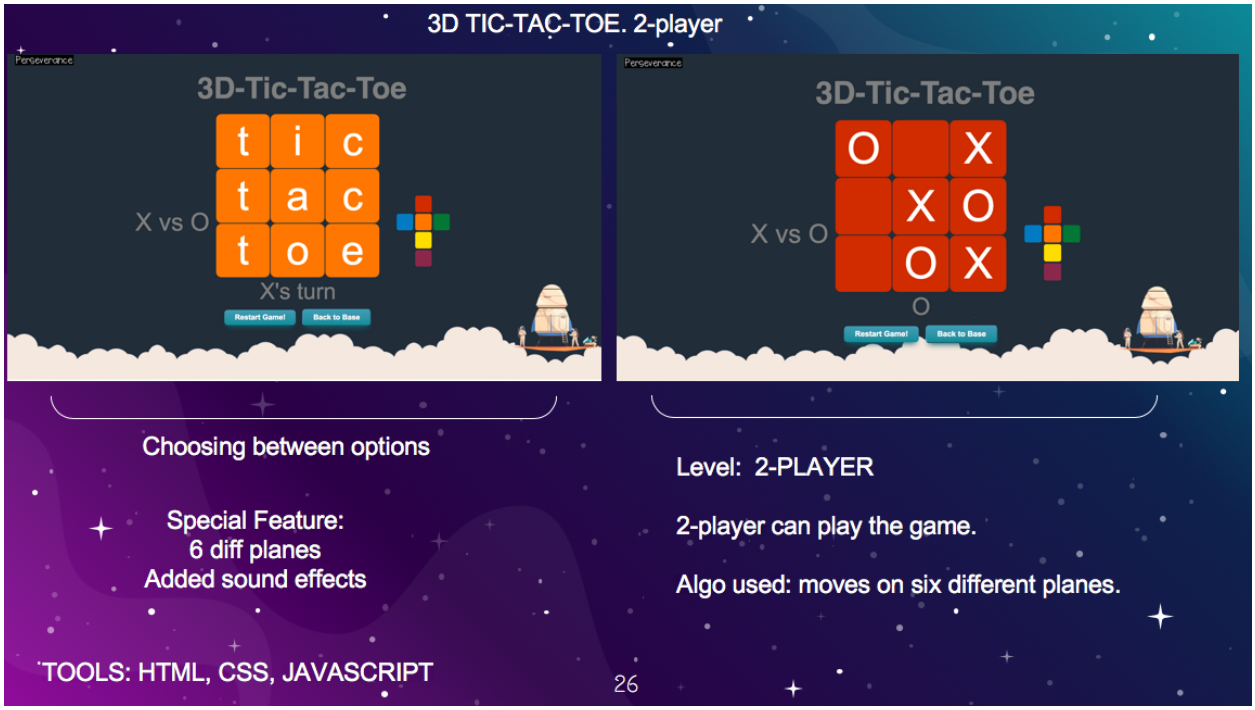
3D LEVEL-2 TIC-TAC-TOE:



3D UNBEATABLE TIC-TAC-TOE:

25

3D 2-PLAYER TIC-TAC-TOE:



26

Future Objective

We are planning to include the following points in our game website.

1) make websites more responsive which can help full users for convenient use of a website.

2) provide database support to our website by which users can login to have a personalized experience on our website may take users name from registration detail and instead of x/o screen will so there names even website may have record of users previous score etc.

3)In this way, we are planning to design another level of our game "Explore the planet" which is using ML algorithms to convert images into 3 by 3 boards of 6 planes of our classical tic-tac-toe and marked safe & unsafe as index value using computer vision algorithms. All these planes are having grid of images taken by satellite of the Mars (in our case may some random image was taken by space open source websites)now player task is to explore all the plains without getting blocked by any plane.s blocking in any plane is similar to the condition of winning in tic tac toe game so we can say this game is an extended version of classical tic tac toe.

Now following the task rules we are thinking of adding in our "explore the planet" game.

1)players have to explore the maximum possible plane as it improves their scores as it can without getting blocked from this site of any plane.
2) After every Move player, the computer using some intelligent algorithm revealed one unsafe index and marked it on that board.
3) if the player is blocked in any of the planes then the game will end.
4) The Player has to move on safer indexes (if it is not it will show a try again message) so that before computers block the plane player explores that plane.
5) Players may save this path as a reference for their future planet exploration tour. In this way explore the planet will become a real-time application game where the crew members along with playing the game will be able to find a safer path for their future exploration of the planet.

Thanks!!!