SRS Document

CS 101 Project:
Quantum Tic Tac Toe

Acknowledgements:

- Prof.**Dr.Deepak B.Phatak**
- Prof.**Dr.Supratik Chakraborty**
- Prof.**Abhiram Ranade**, for his amazingly simple and highly useful **Simplecpp**
- Our CLTA, Miss.**Prerna Gupta**
- **www.countergram.com/qtic** the online game provided us with good visual experience of the game.
- Finally **Google**, it has guided us in many ways

Team Tic Tac Toe:

- Jagadeesh Boddeda-Team Leader
- Uday Kusupati
- Chanukya Vardhan Reddy Gujjula
- Aditya Vardhan Varre

Purpose:

The age old game of classic tic tac toe has evolved in recent times to different forms. One of them is quantum tic tac toe. Quantum tic-tac-toe became the most thoroughly studied abstract quantum system and offered insights that spawned new research. It also turned out to be a fun and engaging game, a game which also provides good pedagogy in the classroom. It helps in making the so called unimaginable and scary quantum physics interesting and also understandable.

Scope:

The game provides a different flavour of a puzzle game not just like one of those crappy number games. The game is reproducible in other higher platforms and with more complexity adding several other features linked with other concepts of quantum physics.

References:

- An Introduction to Programming through C++ - Prof.Abhiram Ranade
- Quantum Tic Tac Toe – Wikipedia
- Quantum Tic Tac Toe – www.countergram.com

Technologies to be used:

- C++ programming language
- s++ compiler – g++ integrated with simplecpp
- Simplecpp graphics library

Product Functions:

The product is a puzzle game that can be played by two individuals or by one with the computer. It is the advanced version of the good old Tic Tac Toe game having implications to quantum physics.
It also includes a Classic Tic Tac Toe game which can be played by two or by only one player with the computer.

Users of the Product:

- Anyone - who needs good amount of brain food
- Students and faculties – for understanding and explaining the various concepts of quantum mechanics

Product Perspective:

The game provides a superior challenge to the puzzle loving gamers .It stands different due to its implications to quantum physics

Software Interface:

- It is possible to play the game in Linux operating environment only due to some limitations in the graphics package.
- s++ compiler(g++ integrated with simplecpp) is required to run the game.

Hardware Interface:

- **Monitor screen** –the software shall display information to the user via the monitor screen
- **Mouse**-the software shall interact with the movement of the mouse and the mouse buttons.The mouse shall activate areas for data input, command buttons

General Constraints:

- Utilization of simplecpp for graphics lays a constraint over the elegancy of the game and also handicaps the game of motion graphics.
- Using brute force computing in some parts of the program reduces the speed and reliability of the program.

User Interface Requirements:

In general, the project has been designed to take inputs from on screen gestures using the mouse. The specific game requirements have been detailed as follows:

The initial interface asks the user to choose in between the two games quantum and classic.

Quantum Tic Tac Toe:

The basis of output is a modified tic tac toe grid on the screen. It is a 3x3 grid with 9 spaces in each square to receive entries. On this grid, as a standard configuration in the beginning of the game, we shall have all the grids and all the spaces within them empty.

The game starts with the instructions appearing on the screen regarding the start of the game by a player. The players play alternately. Two different colours are assigned to the players. The spaces within the grids are highlighted in the colours of the players and also the particle name is indicated in the highlighted space as per the input provide by the player by clicking on the grid.

In the case of loop formation (explained clearly in User Manual), the loop is indicated as a polygon on the screen with the vertices represented as nodes and input for collapse is requested on the screen from the player who formed the loop. The player clicks on one of the grids in which he placed his last particle which is a vertex of the loop so as to collapse his particle over there. All the particles in the

loop are collapsed to their appropriate grids which are coloured depending on the colour of the player who owns the particle.
The program shall display appropriate messages and provide self-explanatory instructions to the user via a suitable chat box, so that the user can play accordingly.

The game ends when either side wins or it is a draw, at which point an appropriate message will be displayed. It may be noted that the graphics library SIMPLECPP has been used in order to implement the graphical user interface of the entire program.

Classic Tic Tac Toe:

The interface initially requires the user to select in between Single player game and two player game.
If single player is chosen, the interface requests the users to choose the difficulty between easy and hard.
It is a normal 3x3 grid. On this grid, as a standard configuration in the beginning of the game, we shall have all the grids and all the spaces within them empty.

The game starts with the instructions appearing on the screen regarding the start of the game by a player. The players play alternately. Two different colours are assigned to the players. The spaces within the grids are highlighted in the colours of the players.(Players may imply two humans or a human and the computer)

The game ends when either side wins or it is a draw, at which point an appropriate message will be displayed. It may be noted that the graphics library SIMPLECPP has been used in order to implement the graphical user interface of the entire program.

Algorithm

Quantum Tic Tac Toe:

-Start.

-Receive input from user regarding which game to play(classic or quantum).

-if quantum,proceed…if,classic open the file of classic tic tac toe using system command.

-Declare a two dimensional array 10x10 A such that it represents the quantum tic tac toe grid (in order to correspond number 1 to 1st grid.)

-Declare a structure named step which contains objects value block 1 and block 2 all of type int

-value indicates the value of each step in a format- tens place represents player name and ones place represents player's move number

-block1 indicates the first block selected by the player in that particular step and similarly block 2.

-Declare two integers points of player1, points of player2 and initialize to 0.

- A[i] 0<i<10 represents individual block of the general tic tac toe

-A[i][0] represents whether the respective block 'i' is collapsed or not.(collapsed if !=0)

-All elements are first initialized to the to 0

- when the player select the un collapsed blocks(since he needs to select two blocks in each turn of him because the position of particle is uncertain and has half probability to be in each of blocks which when collapsed becomes 1)

-check validity and if it is valid (continue reading)

- The first element of the block containing zero (excluding A[][0]) will change to the non-zero value according to the corresponding step of the player .

- if the player1 selects the blocks in his i th step then the value of the first element of the blocks containing zero will change to the value 10+i.(This is the input given) if player2 selects the blocks then it changes to 20+i accordingly.

--after every turn circuit formation needs to be checked (checking circuit condition is discussed in step 2)

-for checking the loop formation using the member objects of structure step i.e value, block 1, block 2 the checkcircuit function is recursed so as to find if a loop is formed.

-Check if circuit is formed by function **checkcircuit**

-If **checkcircuit** returns true and collapse by using function **collapse**; otherwise continue with the next step.

-the structure step is also used in the function collapse.

-Then use the function **checkwin** and if it returns -1 continue

-Otherwise check whether the game is drawn or the player won.

-Output accordingly.

-Stop.

Classic Tic Tac Toe:

Two Player:

-Start

-Declare a one dimensional 10 sized array of type char named square[10].

-square[i] represents i th rectangle of the 3x3 grid.

-All elements are initialised to the value ‘i’ the grid number.

-Input received from the player regarding the selected box is used to update the square[i] value to X or O appropriately.

-The whole square[10] is passed into a function checkwin which returns -1 if the game is not ended,0 if draw, 1 if one of the player wins.

-if it returns 0 or 1 game is over and appropriate message is displayed in the chatbox.

-else input is requested from the next player and the game proceeds.

-Stop.

One Player:

The algorithm for one player game is similar to that of two players

- Instead of receiving input from outside for player two, the computer assigns a random input to the second player
- In some special cases like one of the players is about to win, the computer provides input in such a way that it wins or doesn’t let the other player win.
- For this the compcheck function scans the entire array by placing an extra O or extra X in the empty squares thereby checking if there is a winning move for any of the players. If there is a winning move for any of them the grid no in which the computer has to place its next move is returned.
- Priority is given for the computer’s winning move against the blocking of the player’s winning move.
- If hard difficulty is selected, the computer checks if the first move of the player is in the center of the grid and if so it puts its first move in a corner of the grid.
- It also checks if the center of the grid is vacant in its first move and fills it if

vacant.

- And the other things are same as in two player

Functions and their Specifications:

Quantum Tic Tac Toe:

bool checkCircuit:

-input parameters(by reference)-int A[10][10]

- operates after every turn to find whether a circuit is formed or not.

-In order to check whether circuit is formed we need to back trace indices of blocks which contain numbers present in blocks of last turn which are counterparts of numbers in the selected blocks of last turn(other than numbers involved in last turn).

-Read all elements individually in both blocks

-Return nocircuit if elements other final elements are zeroes in at least one of the blocks otherwise

-Select block with lowest elements and return indices of counterparts (the blocks containing same elements) of each of the elements

-Checking should be carried out similarly.......and if we finally return to the unselected block of final turn then return circuit is formed i.e. returns 'true'

-Back trace follows in such a way that we will select a number in one of the blocks of the final turn and search the blocks containing numbers other the number of last turn and go back to previous blocks and then repeat same procedure.

-If the circuit is formed then it proceeds further to the collapsing of the circuit which enables the player who formed the circuit to fix position of the last particle.

-void collapse :

-input parameters(by reference)-int A[10][10]

-After the circuit is the formed then the player who formed the circuit gets the chance to collapse the circuit

- (i.e. player fixed the position of one particle which results in fixing positions of many other particles.)

-If the (%10 of numbers in the collapsed block) ==1 or 2.

-If 1 then the A[i][0]of counterpart block should be changed in such a way that it represents particle corresponding to player1.A[i][0]=1;if 2 it should represent particle corresponding to player2.A[i][0]=2;

This block is collapsed .Since A [][0] is changed.

-Repeat the same process to collapse the blocks which are counterparts of numbers in the present block.

-Then it becomes a kind of normal tic tac toe type game which has same rules of ***win /loss/draw***.

-for example if block that is collapsed contains 11,21,22 and circuit being completed by 22 then remaining positions of particles corresponding to 11,21 are fixed since only block remains with those respective numbers.so all those blocks are collapsed and process continues till it stops.

int checkWin:

-returns 1 if someone wins, 0 if draw -1 if game is in progress

-Input parameters (by reference)-int A[10][10],int pointsofplayer1,int points of player2

-compare the values of A[][0] .

-The values of A[i][0] represent row1 for i=1,2,3;row2 for i=4,5,6;row3 for i=7,8,9;column1 for i=1,4,7;column2 for i=2,5,8;column3 for i=3,6,9;

Diagonal for i=1,5,9 or i=3,5,7.

-if any row or column or diagonal has same values = 1;then points of player1=points of player1+1.

-similarly for player 2.

-In the function ...takes values of A[i][0] to a separate 3*3 grid and represents values corresponding to player 1 by 1and player 2 by 2.and read every row and column and even diagonals to find whether there are 3 symbols of same player consecutively .if there are three consecutively return player wins. If not found even after all grids are filled then return DRAW....

bool checkInputValidity:

-takes the array A as input and returns true if the input is valid and false otherwise.

-valid if the player plays according to rules (refer User Manual)

Functions pertaining to Graphics:

For the sake of graphic interface of the game the simplecpp.h header file is used in the present project.

All the internal functions in the header file are taken for granted.

void initialInterface:

-initialises the initial tic tac toe board and the name of the game .

void formRectangle:

-takes input four integer parameters.

-takes input the coordinates of centre of rectangle and its dimensions and prints it permanently on the canvas.

bool rectangleCheck:

-takes two integer parameters as input.

-takes input the coordinates of a point and the rectangle number and checks if the point lies in the rectangle

-returns the truth value of its presence

int xCoordinate:

-this function takes an integer parameter which denotes the rectangle number and maps 1,4,7 to 1 and 2,5,8 to 2 and 3,6,9 to 3.

-returns the image of mapping

int yCoordinate:

-this function takes an integer parameter which denotes the rectangle number and maps 1,2,3 to 1 and 4,5,6 to 2 and 7,8,9 to 3.

-returns the image of mapping

void displayInput:

-takes input the tic tac toe array and the coordinates input with the mouse and an integer which represents the player who is playing

-displays the graphic output pertaining to the click of the player and updates the array

Classic Tic Tac Toe:

int AIselect:

-displays the canvas for selection between two player and one player game

-receives input and returns 0 or 1 accordingly.

int levelSelect:

-displays the canvas to choose difficulty for one player game.

-receives input and returns 0 or 1 according to easy or hard.

bool rectangleCheck:

-takes input the coordinates of the input stored in an int in the format specified in simplecpp and a particular rectangle number and returns 1 if the coordinates are within the rectangle else returns 0.

void fillRectangle:

-takes input the move of the player and fills the appropriate rectangle with the appropriate figure square or circle.

int checkWin:

-scans the whole square[10] array and returns 1 if any of the players won.

--1 if none has won yet.

- 0 if game is draw.

int compCheck:

- scans the entire array by placing an extra O or extra X in the empty squares thereby checking if there is a winning move for any of the players.

-If there is a winning move for any of them the grid no in which the computer has to place its next move is returned.

int newGame:

-it is called when the game is over

-it waits until the user clicks in the space allocated to newgame and when it is clicked the function returns 1.

Some functions pertaining to graphics of quantum tic tac toe are also used.