A PROJECT REPORT ON

**2048 GAME**

SUBMITTED IN COMPLETE FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY**

IN

**[INFORMATION TECHNOLOGY]**

Submitted by:

SAMRIDHI KAPOOR 2K20/IT/128

VAISHNAVI 2K20/IT/159

Under the supervision

Of

Dr. Jasraj Meena



Department of Information Technology

Delhi Technological University

Bawana Road Delhi – 110042

NOVEMBER 2021

**DELHI TECHNOLOGICAL UNIVERSITY**

(FORMERLY Delhi College of Engineering)

Bawana Road, Delhi-110042

**CANDIDATE’S DECLARATION**

We (Samridhi Kapoor 2K20/IT/128, Vaishnavi 2K20/IT/159) student of B. Tech. (Information Technology) hereby declare that the Project Dissertation titled "2048 Game” which is submitted to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology is original and not copied from any sources without proper citation, carried out under the supervision of Dr. Jasraj Meena. The work presented in this report has not been submitted and not under consideration for the award for any other course/degree of this or any other Institute/University.

Place: Delhi Samridhi Kapoor (2K20/IT/128)

Vaishnavi (2K20/IT/159)

**INFORMATION TECHNOLOGY**

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY Delhi College of Engineering)

Bawana Road, Delhi-110042

**CERTIFICATE**

I hereby certify that the project Dissertation titled "2048 Game" which is submitted by (Samridhi 2K20/IT/128, Vaishnavi 2K20/IT/159) [Information Technology], Delhi Technological University, Delhi in complete fulfillment of the requirement for the award of the degree of the Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge, the report comprises original work and has not been submitted in part or full for any Course/Degree to this University or else as per the candidate's declaration.

Place: Delhi Dr. Jasraj Meena

(Assistant Professor)

**SUPERVISOR**

**INFORMATION TECHNOLOGY**

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY Delhi College of Engineering)

Bawana Road Delhi-110042

**ABSTRACT**

This 2048 game is built on the idea of combining the same two numbers to make a new number that is the sum of the two numbers. The entire system is written in C++. This is an interesting single-player game. 2048 Game is played on a 4x4 grid with numbered tiles that slide when the four arrow keys are used to move them. Tiles glide in the desired direction until they are stopped by another tile or the grid's edge. Every turn, a new tile will appear at random in an empty place. If two tiles with the same number clash while moving, they will merge into a single tile with the entire value of the two colliding tiles. In the same move, the resultant tile cannot combine with another tile. The user's score starts at zero and is increased by the value of the new tile anytime two tiles join. You can, however, continue to play the game by making greater numbers. A fresh number of displays appears with each step (2 or 4). The player must use the arrow keys to move the numbers and try to collide the same two numbers to form a new number equal to the sum of the two numbers. Numbers slide as far as they can in the desired direction until they hit another number or the grid's edge. Furthermore, the users will find this project to be simple to use and comprehend.

**INFORMATION TECHNOLOGY**

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY Delhi College of Engineering)

Bawana Road, Delhi-110042

**ACKNOWLEDGMENT**

In performing our major project, we had to take the help and guidance of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to show our gratitude to Dr. Jasraj Meena, Mentor for giving us a good guideline for reports throughout numerous consultations.

We would also like to extend our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment. Many people like our classmates and team members themselves, have made valuable comments/ suggestions on this proposal which have inspired us to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

Besides, we would like to thank the Department of Information Technology, Delhi Technological University for allowing us to work on this topic.

**INDEX**

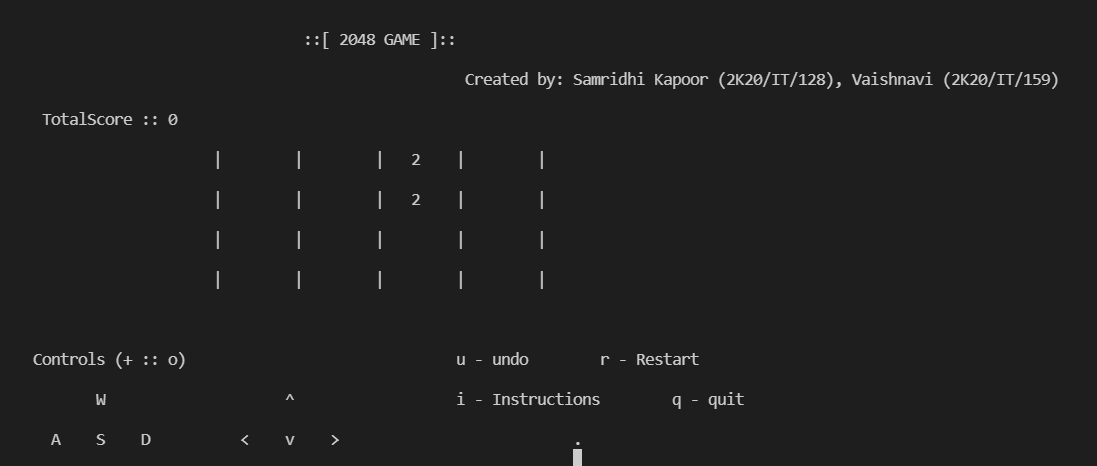
|  |  |  |
| --- | --- | --- |
| Serial No. | Content | Page No. |
| 1. | Title page | 1 |
| 2. | Candidate’s declaration | 2 |
| 3. | Certificate | 3 |
| 4. | Abstract | 4 |
| 5. | Acknowledgment | 5 |
| 6. | Introduction | 7 |
| 7. | How to play | 7-8 |
| 8. | Methodology | 8-17 |
| 9. | Working | 18 |
| 10. | Conclusion | 19 |
| 11. | References | 19 |

**INTRODUCTION**

2048 Game is a single-player video game with sliding tile puzzles. The purpose of the game is to connect numbered tiles on a grid to make a tile with the number 2048; however, after achieving the goal, one can continue to play the game by making tiles with bigger numbers. 2048 is a simple 4x4 grid game with numbered tiles that slide when the four arrow keys are used to move them. Every turn, a new tile with a value of 2 or 4 arrives in a vacant position on the board at random. Tiles slide as far as they can in the specified direction before being stopped by another tile or the grid's edge. If two tiles with the same number clash while moving, they will merge into a single tile with the entire value of the two colliding tiles. In the same move, the resultant tile cannot combine with another tile, the maximum possible tile is 131,072. Only the two tiles furthest along the direction of motion will merge if a move causes three successive tiles of the same value to glide together. If a row or column has all four slots filled with the same value tile, a move parallel to that row or column will merge the first two and last two. The user's score is shown on a scoreboard in the top left corner. The user's score starts at zero and is enhanced by the value of the new tile anytime two tiles join. When a tile with a value of 2048 appears on the board, the game is over. Players can go above and beyond to achieve greater points. The entire code is written in C++ using stacks and arrays data structure. We have created two stacks that would aid in performing the undo operations. One stack is Undo\_Stack that would store elements of the grid in the previous move made by the player and the other is the Score\_Stack that store the scores of the previous move, in case the user wants to perform the undo operation.

**HOW TO PLAY**

* The game starts with 1 or 2 randomly placed numbers in a 4 X 4 grid (16 cells). The rest of the cells are blank.



* To move up, down, left, or right, we must press one of four keys. When we press a key, the elements of the cell move in that direction, so that if there are any two identical numbers in that row (in the case of moving left or right) or column (in the case of moving up and down), they add up, and the extreme cell in that direction fills itself with that number, while the rest of the cells remain empty.

**CONTROLS:**

* W-move up
* A-move left
* D-move right
* S-move down
* After this grid compression any random empty cell gets itself filled with 2 or 4.
* Please don't try to undo consecutively. Maximum 5 undo operations are supported.
* Your objective is to make 2048 in a cell without getting stuck.
* But if during the game there is no empty cell left to be filled with a new 2, then the game goes over. Then it will display lose screen.
* When 2048 is created, the player wins! It will display the following screen. however, continue to play the game by making greater numbers, the maximum possible tile is 131,072.

**NEW FEATURES:**

Username and password

U-undo

R-restart

**METHODOLOGY**

* We have created two 2D arrays of dimensions (4X4) that would be acting as the grids of the 2048 game. One would be the current grid in which the player’s respective moves will be stored, and the other will be a backup grid that stores a move that was just performed before the current move.
* We have created 2 Stacks that would aid in performing the undo operations. One stack is Undo\_Stack that would store the elements of the grid in the previous move made by the user and the other is the Score\_Stack that would store the scores of the previous move, in case the user wants to perform the undo operation.
* We have created a class named Game2048 in which the following functions exist:

class Game2048

{

    stack<vector<vector<int>>> UndoStack;

    stack<int> StackScore;

    string Name;

    int Grid[4][4];

    int CopyGrid[4][4];

    void Initialize();

    void Show();

    void MoveUp();

    void MoveDown();

    void MoveLeft();

    void MoveRight();

    int isFULL();

    int RandomTile(int o);

    void AddUp();

    void AddDown();

    void AddLeft();

    void AddRight();

    void GenerateNewTile();

    int CalMaximum();

    void Instructions();

    int GameOver();

    void DisplayEnd();

    void DisplayWin();

    void DisplayLose();

    void Restart();

public:

    void PlayGame();

    Game2048()

    {

        // DEFAULT CONSTRUCTOR

    }

};

**DESCRIPTION OF THE FUNCTIONS CREATED:**

* + **void Initialize()**

This function initializes the elements of the 2D array. It sets all the elements of the current array and the backup array to 0 initially. Also, it displays two 2s at two random positions in the current array with the help of which the game would start.

void Game2048 ::Initialize()

{

    for (int i = 0; i < 4; i++)

    {

        for (int j = 0; j < 4; j++)

        {

            Grid[i][j] = 0;

            CopyGrid[i][j] = 0;

        }

    }

    int i = RandomTile(4);

    int j = RandomTile(4);

    Grid[i][j] = 2;

    i = RandomTile(4);

    j = RandomTile(4);

    Grid[i][j] = 2;

    Show();

}

* + **void Show()**

This function displays the grid in a tabular form of the 2D array. It displays the options like undo, restart the game, instructions and exit the game to the user.

void Game2048 ::Show()

{

    cout << "\n\t\t\t       ::[ 2048 GAME ]::\n\n";

    cout << "\t\t\t\t\t\t Created by: Samridhi Kapoor (2K20/IT/128), Vaishnavi (2K20/IT/159)\n\n";

    cout << "  TotalScore :: " << TotalScore << endl

         << endl;

    for (int i = 0; i < 4; i++)

    {

        cout << "\t\t     |";

        for (int j = 0; j < 4; j++)

        {

            if (Grid[i][j])

                printf("%4d    |", Grid[i][j]);

            else

                printf("%4c    |", ' ');

        }

        cout << endl

             << endl;

    }

    cout<<"\n\n Controls (+ :: o)\t\t\t\tu - undo\tr - Restart\n\n\tW\t\t     ^\t\t\ti - Instructions\tq - quit\n\t\t\t\t\t\t\t\t"

            <<" \n   A    S    D\t\t<    v    >\t\t\t     ."

            <<" \n\t\t\t\t                             ";

}

* + **void MoveUp()/ MoveDown()/ MoveLeft()/ MoveRight()**

1. This function helps traversing the grid to the bottom of a column.

2. It checks if the tile is empty? If the tile is empty, it further traverses below to find a non-zero element.

3. It then moves the non-zero element to the empty tile.

4. Finally, assign the non-zero element with zero.

void Game2048 ::MoveUp()

{

    for (int i = 0; i < 4; i++)

        for (int j = 0; j < 4; j++)

        {

            if (!Grid[j][i])

            {

                for (int k = j + 1; k < 4; k++)

                    if (Grid[k][i])

                    {

                        Grid[j][i] = Grid[k][i];

                        Grid[k][i] = 0;

                        break;

                    }

            }

        }

}

* + **void isFULL()**

It checks if the grid is full or not. In case any element of the grid is found equal to 0, it returns 0 stating that the grid is not full.

int Game2048 ::isFULL

    int f = 1;

    for (int i = 0; i < 4; i++)

    {

        for (int j = 0; j < 4; j++)

        {

            if (Grid[i][j] == 0)

            {

                f = 0;

                break;

            }

        }

    }

    return f;

}

* + **void DisplayWin()**

It displays the screen when the person has won the game by making a tile equal to 2048.It displays the score the user has obtained and also asks if the user wish to play again.

void Game2048 ::DisplayWin()

{

    char ch;

    cout<<endl<<endl;

    cout<<"\n\t\t\t  :: [  YOU MADE "<<TotalScore<<"!  ] ::"

        <<"\n\n\t\t\t  :: [ YOU WON THE GAME ] ::"

        <<"\n\n\n\n\t\t\t    SCORE\t    NAME";

    printf("\n\n\t\t\t    %6d\t    ",TotalScore);

    cout << "Do you wish to continue???\n";

    cout << "Enter y to continue and n to quit\n\n";

    cin >> ch;

    if (ch == 'n')

    {

        DisplayEnd();

    }

}

* + **void CalMaximum()**

This function traverses the entire grid and finds the maximum element present in it. The value of this maximum element is stored in a variable “max” and is compared again later for the calculation of score.

int Game2048 ::CalMaximum()

{

    int i, j;

    int max = 0;

    for (i = 0; i < 4; i++)

    {

        for (j = 0; j < 4; j++)

        {

            if (Grid[i][j] > max)

            {

                max = Grid[i][j];

            }

        }

    }

    return max;

}

* + **void DisplayEnd()**

This function displays the end screen when the game is over. It displays the user’s final score.

void Game2048 ::DisplayEnd()

{

    cout << "\nYour final Score is :: " << TotalScore << endl

         << endl;

    cout << "THANKS FOR TRYING!!!\n\n"

         << endl;

    exit(0);

}

* + **int GameOver()**

It checks whether the game has ended or not. It does so by traversing the array and checking both row wise and column wise, if 2 adjacent elements are equal or not.

1. If no two adjacent elements are found equal, the game ends. The function returns 1.
2. If two equal adjacent elements are found, the game continues. The function returns 0.

int Game2048 ::GameOver()

{

    int i, j, flag = 1;

    for (i = 0; i < 4; i++)

    {

        for (j = 0; j < 3; j++)

        {

            if (Grid[i][j] == Grid[i][j + 1])

            {

                flag = 0;

                break;

            }

        }

        if (flag == 0)

        {

            break;

        }

    }

    if (flag == 1)

    {

        for (i = 0; i < 3; i++)

        {

            for (j = 0; j < 4; j++)

            {

                if (Grid[i][j] == Grid[i + 1][j])

                {

                    flag = 0;

                    break;

                }

            }

            if (flag == 0)

            {

                break;

            }

        }

    }

    return flag;

}

* + **void GenerateNewTile()**

This function assigns a value of either 2 or 4 to the random indexes. This is done using the rand() and srand() functions.

void Game2048 ::GenerateNewTile()

{

    int flag = 1;

    if (!isFULL())

    {

        while (flag)

        {

            int i = RandomTile(4);

            int j = RandomTile(4);

            if (Grid[i][j] == 0)

            {

                int y = rand() % 10 + 0;

                if (y < 8)

                {

                    Grid[i][j] = 2;

                }

                else

                {

                    Grid[i][j] = 4;

                }

                flag = 0;

            }

        }

    }

}

* + **void AddUp()/AddDown()/AddLeft()/AddRight()**

This function calculates the score obtained after every move. If two adjacent tiles having the same number are merged in a particular move, then it performs the following operation:

**TotalScore += Grid[j][i]**

**UndoScore += Grid[j][i]**

void Game2048 ::AddUp()

{

    for (int i = 0; i < 4; i++)

    {

        for (int j = 0; j < 3; j++)

        {

            if (Grid[j][i] && Grid[j][i] == Grid[j + 1][i])

            {

                Grid[j][i] = Grid[j][i] + Grid[j + 1][i];

                Grid[j + 1][i] = 0;

                TotalScore += Grid[j][i];

                UndoScore += Grid[j][i];

            }

        }

    }

}

* + **void LetsPlay()**

This function takes a character as input from the user as “w”, “a”, “s”, “d” performs the up, left, down a right operation respectively.

void Game2048 ::LetsPlay()

{

    int flag = 0;

    char choice, ch;

    Initialize();

    cin >> choice;

    while ((choice == 'w' || choice == 'a' || choice == 's' || choice == 'd' || choice == 'q' || choice == 'i' || choice == 'u' || choice == 'r'))

If the character input is not “u”, then a 2D array named “current\_copy” of size 4X4 is created, and all the elements of our current grid are pushed into the “current\_copy” using the “push\_back” function. The same process is carried out with another stack called “UndoStack” that would later aid in performing the undo operation.

 if (choice != 'u')

        {

            vector<vector<int>> current\_copy;

            current\_copy.resize(4);

            for (int m = 0; m < 4; m++)

            {

                for (int n = 0; n < 4; n++)

                {

                    current\_copy[m].push\_back(Grid[m][n]);

                }

            }

            UndoStack.push(current\_copy);

        }

Using the switch case operation, we call the below mentioned functions for the respective operations of “w”, “a”, “s” and “d”.

switch (choice){

        case 'w':

            UndoScore = 0;

            MoveUp();

            AddUp();

            MoveUp();

            GenerateNewTile();

            Show();

            StackScore.push(UndoScore);

            break;

for displaying the instructions and restarting the game:

 case 'i':

            Instructions();

            break;

 case 'r':

            Restart();

            break;

**THE UNDO OPERATION:**

If the undo limit is less than 5, and the UndoStack so created is not empty, then create a vector<vector> i.e. a 2d matrix named previous\_copy, and copy the value of UndoStack.top() to the previous\_copy. Perform UndoStack.pop() so as to release the previous \_copy values from indo stack. Lastly, copy the contents of previous\_copy into the Grid[i][j]

The following operation is done on Score:

**TotalScore = TotalScore - StackScore.top()**

Finally, pop out the first element from StackScore() and Increment the undo limit till it reaches 5.

        case 'u':

            if (UndoLimit < 5)

            {

                if (!UndoStack.empty())

                {

                    vector<vector<int>> previous\_copy = UndoStack.top();

                    UndoStack.pop();

                    for (int m = 0; m < 4; m++)

                    {

                        for (int n = 0; n < 4; n++)

                        {

                            Grid[m][n] = previous\_copy[m][n];

                        }

                    }

                    TotalScore -= StackScore.top();

                    StackScore.pop();

                    UndoLimit += 1;

                }

                else

                {

                    cout << "\n\nundo not POSSIBLE, reached initial state!!!\n\n";

                    Show();

                }

            }

            else

            {

                Show();

                cout << "\n\nYou cannot undo the matrix more than 5 times.\n\nSorry!!!\n"

                     << endl;

            }

        }

* + **int main()**

An extra feature of personalized user login with a password protected account is added. If the user enters the wrong password more than three times, the program will stop and exit out of the main function.

int main()

{

    string password = "123";

    string passwordEntry;

    string username;

    int attempts = 1;

    cout<<"\n\t\t\t..........WELCOME..........\n";

    cout<<"\n\t\t\t     ::[ 2048 Game ]::\n"<<endl;

    cout<<"\t\t\tPlease enter your Username: ";

    cin>>username;

    cout << "\t\t\tPlease enter your password: ";

    cin>>passwordEntry;

    cout<<endl;

    while ( passwordEntry != password && attempts <=2 )

    {

        cout << "Please try again: ";

        cin>>passwordEntry;

        cout<<endl;

        attempts++;

    }

    if ( passwordEntry == password && attempts <=3 )

    {

        cout << "\n\t\t\t\tACCESS GRANTED!\n\n\t\t------------------------------------------------\n\n";

    }

    else

    {

        cout << "Sorry, only 3 attempts are allowed!";

        exit(0);

    }

    Game2048 p;

    srand(time(NULL));

    p.LetsPlay();

    return 0;

}

**WORKING**

1. **ESSENTIAL VARIABLES/FUNCTIONS**

We require 2 grids, one that keeps the current state of the game (current grid) and another that keeps the previous state, i.e., a move before (backup grid). The backup grid serves 2 purposes:

* 1. For the UNDO functionality, resuming to the previous state.

1.2 To check if the grid moves.

Random function to generate random positions (rows and columns) for the initial two tiles on the board and for the further tiles which will be generated on each move. The thing to note here is that probability of occurrence of 4 is less than that of 2. Greatest function to find the greatest tile in the grid at any state.

1. **UNDO**

To perform UNDO operation, copy the backup grid to the current grid.

1. **UPDATING THE GRID**

* Check if adjacent tile is equal.
* Sum / double the tile(s) if they are equal.

1. **SPAWNING A NEW TILE**

With every move, a new tile should be spawned in a random position under the condition that the selected direction moves at least 1 tile in the grid. So, this is where the backup grid comes into use, before updating the grid. Check if the current grid is equal to the backup grid. So that we know if at least one tile in the grid has moved. Then spawn a new tile in an empty cell.

1. **RANDOM GENERATION**

Make the computer play the game on its own! For that, remove the input call and create a char array with all the four direction keys (W, A, S, D) and choose a key in random. But if you’ve played 2048 long enough, you’d know that skipping a direction is the best strategy, so choose any 3 directions.

1. **THE GAME ENDS WHEN**

* Grid is full and a new tile cannot be spawned.
* The game is won, 2048 has been created.

To check whether the grid is full, check if all the elements are non-zero.

**CONCLUSION**

Game 2048 is a popular single-player video game released by Italian programmer Gabriele Cirulli. Specifically, the game is played on a 4 x 4 grid board, with each tile holding an even number. Player can swipe on the board in four directions: up, down, left and right; then all tiles will move in that direction until stopped by another tile or the border of the grid. If two tiles of the same number collide while moving, they will merge into a new tile bearing their summation. The new tile cannot merge with another neighbor tile again during this move. After the move, a new tile with number 2 or 4 will randomly appear on one of the empty tiles, and then, player makes a new move. When there is no empty cell and no more valid move, the game ends. If a tile with number 2048 appears in the board before the game ends, the player wins. A scoreboard keeps track of the user’s score. The user’s score starts at zero, and is incremented whenever two tiles combine, by the value of the new tile. As with many arcade games, the user’s best score is shown alongside the current score

**REFERENCES**

1. https://www.geeksforgeeks.org/2048-game-in-python/

2. https://play2048.co/

3. <https://en.wikipedia.org/wiki/2048_(video_game)>

**LINK TO THE CODE**

<https://github.com/samridhikapoor/2048-game-in-cpp>