

Sam Dyas 26011251  
“Valley of The Kings”  
Game Project Report

# Introduction

For my final project I produced a text-based puzzle/adventure game using C++. I chose to use C++ as user input is simpler to code using 'Cin' and 'Cout' as opposed to in regular C.

There are five puzzles to solve and four different endings/achievements to unlock.

There are approximately 894 lines of code.



# 1. Boolean Statements

To switch between 'levels' within the game boolean true/false statements are used. As shown below, when 'inmenu == true' the functions of the Main Menu are executed. If the first option to start the game is chosen by the user, the statement is declared false and the main menu is exited, moving onto the subsequent code (not shown) to start the game. All levels are initially declared false and are declared true when appropriate.

```
while (gameon = true)
{
    bool predoorpuzzle = false; //Introduction
    bool doorpuzzle = false; //DoorPuzzle
    bool doorpuzzlepathA = false; //Chooses Asp
    bool doorpuzzlepathB = false; //Chooses Tomb
    bool waterpuzzletext = false; //Water Puzzle
    bool mazepuzzle = false; //MazePuzzle
    bool mazepuzzlechoice = false;
    bool finalpuzzle = false; //FinalPuzzle
    bool inmenu = true; //MainMenu
    while (inmenu == true)
    {
        menutext(); //Displays Main Menu text
        char menuinput;
        cin >> menuinput;
        switch (menuinput)
        {
            case ('1'): //Declares inmenu false and starts game
                inmenu = false;
                break;
            case ('2'): //Displays Achievements then returns to Main Menu
                system("CLS");
        }
    }
}
```

## 2. Functions

To tidy up the code within the main function and to make it easier to read, large blocks of text are contained as functions in the header files and then called when appropriate. Each puzzle has an individual header file containing relevant functions.

The function below displays the Main Menu text and is found within the menutext header file.

```
void menutext() //Main Menu Screen
{
    system("CLS");
    cout << "Valley of The Kings" << endl;
    cout << "XXXXXXXXXXXXXXXXXXXX" << endl;
    cout << "1. Start Game" << endl;
    cout << "2. Achievements" << endl;
    cout << "3. Credits" << endl;
    cout << "XXXXXXXXXXXXXXXXXXXX" << endl;
}
```

# 3. Arrays and Conditional Statements

There are four achievements to unlock while playing the game. Each achievement is displayed using arrays. Each array entry contains text, initially showing the achievement is locked, and subsequently unlocked after certain conditional statements are met.

The code on the right shows: the declaration, the output of the arrays within the main menu, and a conditional boolean statement to unlock an achievement.

The achievements are declared separately to the rest of the game code, which is found within the 'while (gameon==true)' section of the main . This is so after an ending is met and the player is returned to the main menu, the achievements are not redeclared as false.

```
int main()
{
    string achievement[5]; //Declares Achievements Array (all initially locked)
    achievement[0] = "Ending 1: Locked";
    achievement[1] = "Ending 2: Locked";
    achievement[2] = "Ending 3: Locked";
    achievement[3] = "Ending 4: Locked";

    bool gameon = true;

    while (gameon = true)
    {...

case ('2'): //Displays Achievements then returns to Main Menu
    system("CLS");
    cout << "Achievements:" << endl;
    cout << achievement[0] << endl;
    cout << achievement[1] << endl;
    cout << achievement[2] << endl;
    cout << achievement[3] << "\n" << endl;
    system("PAUSE");
    break;

else if (snakepuzzlecomplete == false)
    {
        achievement[1] = "Ending 2: Had To Be Snakes";
        doorpuzzlepathA = false;
        inmenu = true;
        break;
    }
```

# 4. Switch/Case

Switch and Case are used to navigate the main menu options, this is the only instance in which it is used; boolean statements are used for navigation between levels.

```
while (inmenu == true)
{
    menutext(); //Displays Main Menu text
    char menuinput;
    cin >> menuinput;
    switch (menuinput)
    {
        case ('1'): //Declares inmenu false and starts game
            inmenu = false;
            break;
        case ('2'): //Displays Achievements then returns to Main Menu
            system("CLS");
            cout << "Achievements:" << endl;
            cout << achievement[0] << endl;
            cout << achievement[1] << endl;
            cout << achievement[2] << endl;
            cout << achievement[3] << "\n" << endl;
            system("PAUSE");
            break;
        case ('3'): //Displays Credits then returns to Main Menu
            menuchoice3();
            break;
    }
}
```

# Puzzles

There are five puzzles to be solved. The Door and Snake puzzles use simple conditional choices, the former with a word search. The Water and Sarcophagus puzzles involve changing vectors to the correct size. The fourth puzzle requires the user to follow directions through a maze.



# Vector Puzzles

Both the third and fifth puzzles require the user to change the relevant vectors to the correct sizes, as inferred from text statements displayed with each puzzle.

As shown in the water puzzle code on the right: the user must select the vector (container) and change the size. A basic value of 1 is pushed back to the vector with iterations according to the user input. The user can empty the vectors if a mistake is made.

Once the vectors are all the correct size the puzzle is complete.

```
char containerchoice;
cin >> containerchoice;
if (containerchoice == '1')
{
    cout << "\nHow many litres of water will you fill it with?\n" << endl;
    // Fills first container
    cin >> volume1;
    for (int i = 0; i < volume1; i++) watercontainer1.push_back(1);
}
if (containerchoice == '2')
{
    cout << "\nHow many litres of water will you fill it with?\n" << endl;
    // Fills second container
    cin >> volume2;
    for (int i = 0; i < volume2; i++) watercontainer2.push_back(1);
}
if (containerchoice == '3')
{
    cout << "\nHow many litres of water will you fill it with?\n" << endl;
    // Fills third container
    cin >> volume3;
    for (int i = 0; i < volume3; i++) watercontainer3.push_back(1);
}
if (containerchoice == '4') // Empties containers of water
{
    watercontainer1.clear();
    watercontainer2.clear();
    watercontainer3.clear();
}
system("CLS");
if ((int)watercontainer1.size() == 3) // Correct Container Volume 1
{
    cout << "The first pressure plate activates." << endl;
}
if ((int)watercontainer2.size() == 21) // Correct Container Volume 2
{
    cout << "The second pressure plate activates." << endl;
}
if ((int)watercontainer3.size() == 7) // Correct Container Volume 3
{
    cout << "The third pressure plate activates." << endl;
}
if ((int)watercontainer1.size() == 3 && (int)watercontainer2.size() == 21 &&
(int)watercontainer3.size() == 7)
{
    waterpuzzleactive = false;
}
```

# Maze Puzzle

The fourth puzzle requires the user to navigate a maze. In each room of the maze the user is given directions, if the wrong direction is followed they return to the first room and the 'lost counter' vector increases size by one. If the player takes the wrong direction three times they become fully lost and reach an ending state. Otherwise if all rooms are navigated successfully they leave the maze and transition to the next level.

```
while (room6 == true)
{
    system("CLS");
    cout << "The wooden sign in the middle of the room reads: Go in the
direction you went two rooms ago\n" << endl; // Right
    directions();
    char roomchoice6;
    cin >> roomchoice6;
    if ((roomchoice6 == '1') || (roomchoice6 == '2')) // Back to Room 1
    {
        roomreturn1();
        if ((int)lostcounter.size() == 2)
        {
            losttext();
            mazepuzzleactive = false;
            room6 = false;
            break;
        }
        roomreturn2();
        room6 = false;
        room1 = true;
    }
    else if (roomchoice6 == '3') // Exit Maze
    {
        system("CLS");
        cout << "With some luck you manage to navigate to the other end
of the maze.\n" << endl;
        system("PAUSE");
        mazecomplete = true;
        mazepuzzleactive = false;
        room6 = false;
    }
}
if ((int)lostcounter.size() == 2) // If lost 3 times ENDING 3
{
    break;
}
if (mazecomplete == true) // Maze Solved
{
    break;
}
```

# Development in Relation To The Project Design Document

The Project Design Document lists the requirements I thought would be necessary for the game:

- Must have different options for players to select
- Should have multiple pathways and endings to encourage replay
- Could have a scoring system (collectibles)
- Must have puzzles of increasing difficulty
- Should have items to pick up (determine pathways/endings)
- Should have images of rooms/items/characters

Of these requirements: the scoring system, items, and images of rooms were not implemented.

I could not think of how to implement a scoring system seeing as the puzzles were not timed, as such the only 'score system' of sorts is the achievements list. This occupies the same role as a score system, giving the player an incentive to replay and complete all of the game's endings. The items were unnecessary for players to collect. I chose to plan the pathways according to player decisions during the puzzles. Adding items would have made it more complicated to code. Lastly I did not create images of the rooms/levels due to lacking the artistic ability. Adding the art would have been nice however minimal was lost by it's exclusion.

I did however implement: multiple choices, endings, and puzzles of increasing difficulty. There are four endings, of which three are failure states, and the fourth the successful completion of the game. Furthermore the player's path and ending is determined by the choices they make for the puzzles. Unlocking all endings requires four play-throughs of the game

# Testing

Due to the text based nature of the game I had to conduct few tests. I developed each section of the game individually and tested as I went along to make sure everything up to that point worked. I first created the main menu, and subsequently each puzzle.

Containing each puzzle within a separate header file allowed me to easily access and edit the code in isolation without having to search through hundreds of lines, as well as allowing me to test everything within a particular section thoroughly before moving onto the next section.

I tested the following as I went along, and upon completing the project:

1. User Input
2. Level Transition and Level Selection
3. Vector Puzzle Input
4. Achievement Unlocks

# 1. User Input

Each puzzle relies on the user inputting a number to select an option or quantity. Initially I used 'int' values however inputting a non-integer character would result in the game bugging out. Thus, I replaced it with 'char' inputs. This fixed the error.

## 2. Level Transition and Level Selection

There are five consecutive puzzles/levels within the game. When the completion condition for the current level is met, the boolean statement for that level is declared false, and the statement for the following level is declared true. This allows for transition between the levels.

Upon adding a new level I ran through the previous level to make sure the new transition worked as intended. Additionally I created a level select option in the main menu. This allowed me to skip parts of the game that had already been tested, saving time. It also allows the player to skip parts of the game they have already completed, when replaying the game.

(Right, Level Selection)

```
developerchoice(); // Level Select/Developer Options
char developerchoice;
cin >> developerchoice;
if (developerchoice == '1')
{
    developermode();
    char levelselect;
    cin >> levelselect;
    if (levelselect == '1')
    {
        system("CLS");
        doorpuzzle = true;
    }
    else if (levelselect == '2')
    {
        system("CLS");
        doorpuzzlepathA = true;
    }
    else if (levelselect == '3')
    {
        system("CLS");
        doorpuzzlepathB = true;
    }
    else if (levelselect == '4')
    {
        system("CLS");
        mazepuzzle = true;
    }
    else if (levelselect == '5')
    {
        system("CLS");
        finalpuzzle = true;
    }
    else if (levelselect == '6')
    {
        system("CLS");
        inmenu = true;
    }
}
```

# 3. Vector Puzzle Input

The third and fifth puzzle rely on vector size. I had to ensure the correct vector sizes met the requirements of the conditional statements. This was simply tested by inputting the required values to display the completion text for each puzzle. If the player inputs incorrect values they can reset the puzzle and retry.

The vector sizes are reset to zero upon transitioning to the next level so the player can re-attempt the puzzle during replays.

(Below, conditional statement for fifth puzzle)

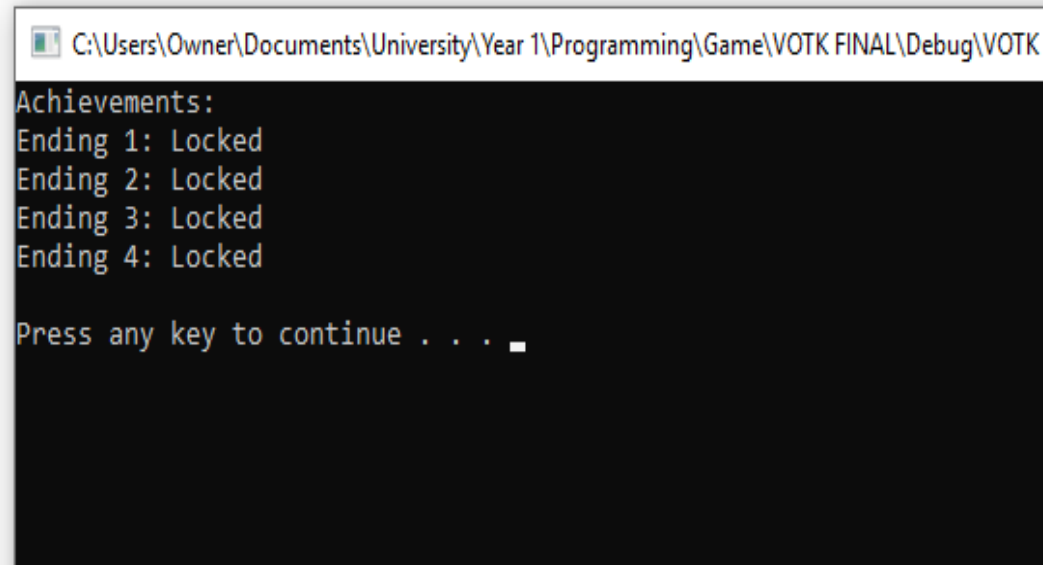
```
if ((int)switch2.size() == 6 && (int)switch3.size() == 10 && (int)switch4.size() == 50 &&
(int)switch5.size() == 21)
{
    switchpuzzleactive = false;
    break;
}
```

# 4. Achievements

Each achievement must display the locked and unlocked state. The player must be able to achieve all endings and unlock all achievements.

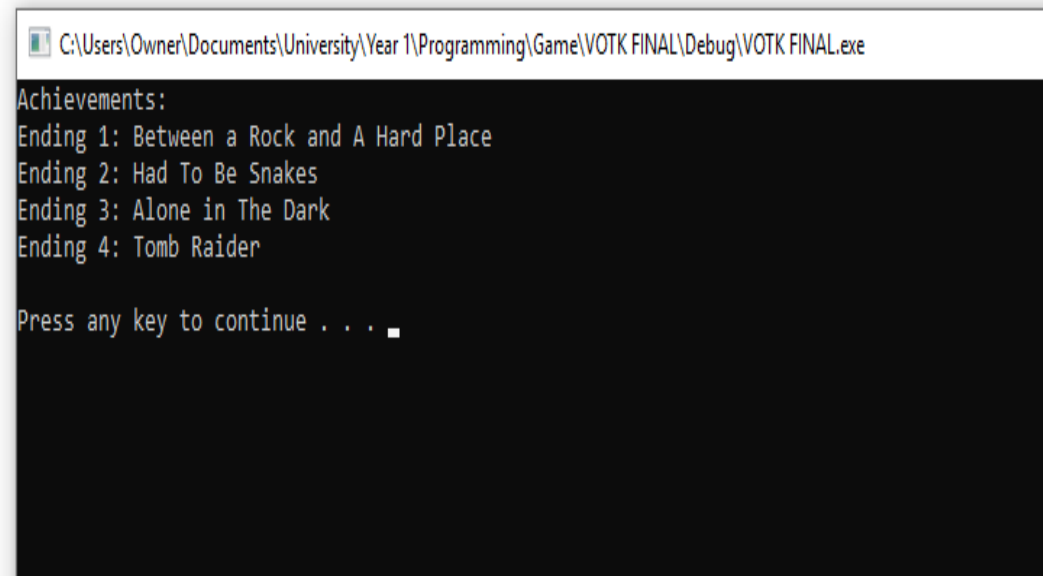
As such I played through each path to ensure upon completing each ending, the player is returned to the main menu, and the achievement is then shown as unlocked.

The system works as intended and the player can unlock all achievements.



```
C:\Users\Owner\Documents\University\Year 1\Programming\Game\VOTK FINAL\Debug\VOTK
Achievements:
Ending 1: Locked
Ending 2: Locked
Ending 3: Locked
Ending 4: Locked

Press any key to continue . . . .
```



```
C:\Users\Owner\Documents\University\Year 1\Programming\Game\VOTK FINAL\Debug\VOTK FINAL.exe
Achievements:
Ending 1: Between a Rock and A Hard Place
Ending 2: Had To Be Snakes
Ending 3: Alone in The Dark
Ending 4: Tomb Raider

Press any key to continue . . . .
```



# Conclusions and Review

This final project has been a great way for me to consolidate the knowledge acquired during my first year at university. I came in with very minimal programming experience and while occasionally struggling during some lab practicals, I have ended the year with the confidence to code independently and produce work I can be proud of. I have enjoyed the freedom to decide how I approach the project and believe it has allowed me to produce work which may be of a higher quality than if it had had to follow stricter specifications. I have been able to revisit challenging areas such as arrays and vectors, and have implemented them into my game. While no expert, I feel comfortable coding in C++ and through the project and the lab practicals I have witnessed the many ways in which it can be used to help in the workplace or when producing a product, such as databases or games.

I have been provided with a strong foundation on which to improve my knowledge and understanding of coding and to continue my studies into the second and third years. It must be said that the game project has most likely been my favourite aspect of the year, not least due to the freedom to develop it to my liking, as well as the ample time to complete it in. Still, I found myself confused and frustrated at certain points during which the code wouldn't work. Naturally, these obstacles were overcome with periods of intense Googling and intense sleep, perhaps both signs I am destined to be a programmer.