Greg Balbirnie
1500405

COMPUTER GAMES ARCHITECTURE

- I decided to recreate the hacking minigame from Fallout 4 in which the player has to figure out the correct password. When the player selects a word they will be told how many letters are the same as in the password and from that can figure out the password. The game will end either when the player selects the password or after three incorrect selections.

Main Function.

Sets up the backgrounds, palette, and charblocks that are used in the program.

```
//main function
int main()
{
    // Set display options.
    REG_DISPCNT = DCNT_MODE0 | DCNT_BG0 | DCNT_BG1 | DCNT_BG2;

    // Set background options.
    REG_BG2CNT = BG_CBB(0) | BG_SBB(30) | BG_8BPP | BG_REG_32x32;
    REG_BG0HOFS = 0;
    REG_BG0VOFS = 0;
    REG_BG1CNT = BG_CBB(0) | BG_SBB(29) | BG_8BPP | BG_REG_32x32;
    REG_BG1HOFS = 0;
    REG_BG1VOFS = 0;
    REG_BG0CNT = BG_CBB(1) | BG_SBB(28) | BG_8BPP | BG_REG_32x32;
    REG_BG2HOFS = 0;
    REG_BG2VOFS = 0;

    // Set up the palette.
    SetPaletteBG(0, RGB(0, 0, 0)); // black
    SetPaletteBG(1, RGB(0, 27, 0)); // pale green
    SetPaletteBG(2, RGB(0, 7, 0));; // dark green

    //load the font
    for (int x = 0; x < 128; x++)
    {
        LoadTile8(0, x, font_medium[x]);
    }

    //set the dark green tile
    LoadTile8(0, 1, green_tile);

    //set the life tile
    LoadTile8(0, 2, life_tile);

    //load the inverted font
    for (int x = 0; x < 128; x++)
    {
        LoadTile8(1, x, font_bold[x]);
    }

    // make screenblock 30 green.
    for (int y = 0; y < 32; ++y)
    {
        for (int x = 0; x < 32; ++x)
        {
            SetTile(30, x, y, 1);
        }
    }

    //Write the aesthetic stuff up the top
    DrawText(29, 0, 0, "C:\\WEBCORP");

    LevelSelect();

}
```

Level_Select Function.

Allows the player to select the level they wish to play and sets each level up.

```
//pressing A (Z on keyboard)
if ((currentKeys & KEY_A) == 0 && (prevKeys & KEY_A) != 0)
{
    switch (pointer)
    {
    case 0: words[0] = "BRAID";
            words[1] = "BREAD";
            words[2] = "GRATE";
            words[3] = "CHAIN";
            words[4] = "CARDS";
            words[5] = "STARS";
            words[6] = "DREAM";
            words[7] = "DOORS";
            words[8] = "TOWEL";
            words[9] = "PRONG";
            ClearScreen();
            Game();
            break;
    case 1: words[0] = "SATIN";
            words[1] = "TRAIN";
            words[2] = "STAIN";
            words[3] = "CHALK";
            words[4] = "CHESS";
            words[5] = "TREES";
            words[6] = "MESSY";
            words[7] = "TESTS";
            words[8] = "TRACK";
            words[9] = "TIMES";
            ClearScreen();
            Game();
            break;
    case 2: words[0] = "SPOOK";
            words[1] = "CLAWS";
            words[2] = "MERGE";
```

Game Function.

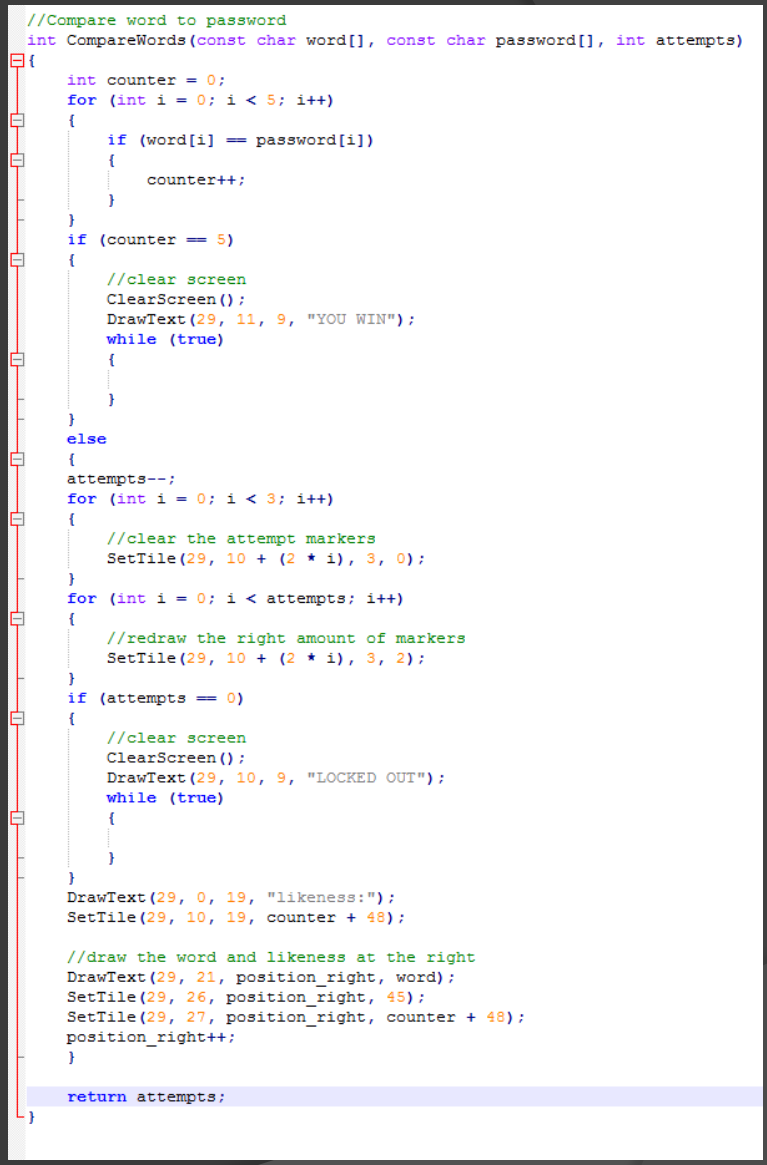
Puts the tiles on screen and allows the user to play the game.

```
srand(time(NULL));
//variables
int attempts = 3;
int positions[10];
string password;
int rando;

//choose password
rando = rand() % 9;
password = words[rando];

//fill screen with random symbols
for (int y = 6; y < 18; y++)
{
    for (int x = 0; x < 20; x++)
    {
        rando = rand() % 14;
        rando = (rando + 33);
        SetTile(29, x, y, rando);
    }
}

//Write the aesthetic stuff up the top
DrawText(29, 0, 0, "C:\\WEBCORP\\PASSCODE\\");

DrawText(29, 0, 3, "ATTEMPTS:");

//print the words
for (int i = 0; i < 10; i++)
{
    rando = rand() % 15;
    positions[i]= rando;
    DrawText(29, rando, i + 7, words[i].c_str());
}

//prepare the main loop
int pointer = 0;
uint16_t prevKeys = 0;
DrawText(28, positions[pointer], pointer + 7, words[pointer].c_str());
DrawText(29, 21, 6, words[pointer].c_str());
for (int i = 0; i < attempts; i++)
{
    SetTile(29, 10 + (2 * i), 3, 2);
}
```

```
uint16_t currentKeys = REG_KEYINPUT;

//pressing down
if ((currentKeys & KEY_DOWN) == 0 && (prevKeys & KEY_DOWN) != 0)
{
    pointer++;
    if (pointer > 9)
    {
        pointer = 9;
    }

    // make screenblock 28 blank
    for (int y = 0; y < 32; ++y)
    {
        for (int x = 0; x < 32; ++x)
        {
            SetTile(28, x, y, 0);
        }
    }

    DrawText(28, positions[pointer], pointer + 7, words[pointer].c_str());
    DrawText(29, 21, 6, words[pointer].c_str());
}

//pressing up
if ((currentKeys & KEY_UP) == 0 && (prevKeys & KEY_UP) != 0)
{
    pointer--;
    if (pointer < 0)
    {
        pointer = 0;
    }

    // make screenblock 28 blank
    for (int y = 0; y < 32; ++y)
    {
        for (int x = 0; x < 32; ++x)
        {
            SetTile(28, x, y, 0);
```

Compare_Words Function.

Compares each letter in the selected word and the password and checks whether the player has won or not.



```
//Compare word to password
int CompareWords(const char word[], const char password[], int attempts)
{
    int counter = 0;
    for (int i = 0; i < 5; i++)
    {
        if (word[i] == password[i])
        {
            counter++;
        }
    }
    if (counter == 5)
    {
        //clear screen
        ClearScreen();
        DrawText(29, 11, 9, "YOU WIN");
        while (true)
        {

        }
    }
    else
    {
    attempts--;
    for (int i = 0; i < 3; i++)
    {
        //clear the attempt markers
        SetTile(29, 10 + (2 * i), 3, 0);
    }
    for (int i = 0; i < attempts; i++)
    {
        //redraw the right amount of markers
        SetTile(29, 10 + (2 * i), 3, 2);
    }
    if (attempts == 0)
    {
        //clear screen
        ClearScreen();
        DrawText(29, 10, 9, "LOCKED OUT");
        while (true)
        {

        }
    }
    DrawText(29, 0, 19, "likeness:");
    SetTile(29, 10, 19, counter + 48);

    //draw the word and likeness at the right
    DrawText(29, 21, position_right, word);
    SetTile(29, 26, position_right, 45);
    SetTile(29, 27, position_right, counter + 48);
    position_right++;
    }

    return attempts;
}
```

- I used the GBA buttons to let the player cycle through the words on screen as well as select the word they think is the password.
- I displayed this using tiles of the fonts declared in the fonts file.
- I used three layers that were edited whilst the program was running.
- I used the randomise function to display characters in different places.
- I used arrays to store the words that will be displayed