Research Document

Narrative - (Technical Storytelling):

The user (James) opens the game window -> James sees a character sprite of a warrior, a word YELLOW and a word RED -> The words begin to move from right to left across the screen -> James presses the [right arrow] key -> The main character sprite on the screen moves right -> James presses the [Spacebar] and notices the main character sprite intersects with the word YELLOW -> The screen turns yellow.

Process - (Example of some of the raw code and a step through set of comments explaining how it works):

Code Responsible for moving the YELLOW and RED word sprites:

```
void Game::moveSprite()
{
    short count = 0;

    while (count <= 3200)
    {
        m_yellowOptionS.setPosition(priorY - 0.1f, 120.0f);
        priorY = priorY - 0.1f;

        m_redOptionS.setPosition(priorR - 0.1f, 240.0f);
        priorR = priorR - 0.1f;

        count++;
    }
```

Code Responsible for allowing the player to move the main character sprite:

```
void Game::moveSprite()
{
    short count = 0;
    /*
    while (count <= 3200)
    {*/
    if (count <= 3200)
    {

        m_yellowOptionS.setPosition(priorY, 120.0f);
        priorY = priorY - 0.1f;

        m_redOptionS.setPosition(priorR, 240.0f);
        priorR = priorR - 0.1f;

        count++;
    }
```

Research Document

Code Responsible for checking if sprites collide and outputting a message to confirm check has been done correctly:

```
void Game::checkCollision()
{
    if ( m_mainCharS.getGlobalBounds().intersects( m_yellowOptionS.getGlobalBounds()))
    {
        m_switchColourY = true; //For testing
        m_colour = colourMode::yellow;
    }

    if ( m_mainCharS.getGlobalBounds().intersects( m_redOptionS.getGlobalBounds()))
    {
        m_switchColourR = true; //For testing
        m_colour = colourMode::red;
    }

    if (m_switchColourY == true) //testing message
    {
        std::cout << "Y = true" << std::endl;
        m_switchColourY = false;
    }
    if (m_switchColourR == true) //testing message
    {
        std::cout << "R = true" << std::endl;
        m_switchColourR = false;
    }
}
```

Code Responsible for changing screen colour:

```
void Game::changeColour()
{
    //to set screen colour when collison occurs
    switch (m_colour)
    {
    case colourMode::black:
        m_window.clear(sf::Color::Color(127, 127, 127, 255));
        break;
    case colourMode::yellow:
        m_window.clear(sf::Color::Yellow);
        break;
    case colourMode::red:
        m_window.clear(sf::Color::Red);
        break;
    }
```

Image - (Clip of what the layout of the project in the terminal/game window actually looks like):