CSC 1030 Group Project: Individual Contribution

We worked well as a group and everyone was involved in everything to some extent. Each of us took turns to pull it all together and tidy up any mistakes we noticed. The bits I had most contribution to were as follows:

1. I created the plot and conceptualised the pages with Alexander.
2. I decided on the graphics & implemented/ensured continuity in the CSS throughout.
3. I made background graphics using our colour scheme.
4. I created the first page name enter form & implementation of this personalisation feature throughout the rest of the game. Using HTML form to gather data on the user and then using localStorage javascript feature to hold this data and then fillName( ); method to retrieve it throughout the game.
5. I made a global countdown timer for the game to ensure gameplay was stopped after 5mins/300s. This used javascript setInterval and local storage to store the remaining time at the end of each page and get it on the next page and continue the countdown.
6. I developed the dog attack game.
7. I the made countdown timer circle display and the methods that this calls on timeout.
8. I linked all the pages together & made sure they work together at various stages.
9. I added Help/restart button to every page & adjusted code to ensure continuity.
10. I added easy-read feature to all pages and adjusted all css to respond to greater font size.

Examples of some of these features on the next pages…

Countdown Timer text & visual display

David Montgomery

Individual Contribution

CSS

```css
.backgroundCircle {
  position: relative;
  width: 200px;
  height: 200px;
}

.backgroundCircleSvg {
  -webkit-transform: scaleX(-1);
      -ms-transform: scaleX(-1);
          transform: scaleX(-1);
}

.foregroundCircle {
  fill: none;
  stroke: none;
}

.foregroundElapsed {
  stroke-width: 10px;
  stroke: black;
}

.foregroundRemaining {
  stroke-width: 7px;
  stroke-linecap: round;
  -webkit-transform: rotate(90deg);
      -ms-transform: rotate(90deg);
          transform: rotate(90deg);
  -webkit-transform-origin: center;
      -ms-transform-origin: center;
          transform-origin: center;
```

Result of timeout on Dogfight level

CSS for countdown timer: some of the code to create the circle representation of time counting down, which changes colour through orange and filling the circle, through orange and red as the time remaining gets closer to 0. I also added a single beat effect on the time display which begins when half your time is up, which then progresses to a double 'heartbeat' like effect in the last 5 seconds, as a visual representation of the urgency & to help create an engaging and immersive gameplay.

HTML for countdown timer: Keeping it simple to one single div to allow easy insertion in other pages by other members of the team.

```html
<!DOCTYPE HTML>
<html>
<head>
  <title>iSPY - Countdown</title>
  <link rel="stylesheet" href="countdownCircle.css">
  <link href='https://fonts.googleapis.com/css?family=Orbitron' rel='stylesheet' type='text/
</head>
<body>
<div id="timerDisplay"></div>
<script type="text/javascript" src="countdownCircle.js"></script>
</body>
</html>
```

Javascript for countdown timer: an example of how I used Javascript to countdown in seconds, change the timer display circle from white, through orange and red in proportion to the remaining time, pop/beat the text towards the end of the time limit, and on timeout stop & collapse the timer, set the player's health to 0%, and display the 'MISSION FAILED' div.

Javascript

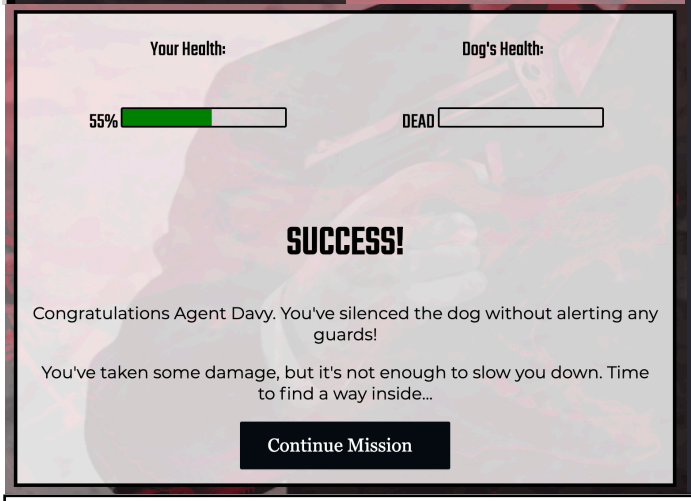
```javascript
function startTimer() {
    var sec = 30;
    var timer = setInterval(function(){
        sec--;
        if (timerShouldStop) {
          clearInterval(timer);
        }
        else if (sec < 0) {
            clearInterval(timer);
            causeOfDeath = "Dog alerted the Guards";
            localStorage.setItem("causeOfDeath", causeOfDeath);

            playerHealth = 0;
            playerHealthValue.style.width =  playerHealth + "%";
            playerHealthLabel.innerHTML =  playerHealth + "%";
            document.getElementById('dogIntro').style.display = 'none';
            document.getElementById('attack').style.display = 'none';
            document.getElementById('lastAttack').style.display = 'none';
            document.getElementById('lostFight').style.display = 'block';
        }
    }, 1000);
}
```

```javascript
//format time to min:sec
function formatTime(time) {
  const minutes = Math.floor(time / 60);
  let seconds = time % 60;

  if (seconds < 10) {
    seconds = `0${seconds}`;
  }

  return `${minutes}:${seconds}`;
}

//set foreground circle colour and make text pop when thresholds reached
function setRemainingPathColor(timeLeft) {
  const { alert, warning, initial } = COLOR_CODES;
  if (timeLeft <= alert.threshold) {
    document.getElementById("foregroundRemaining").classList.remove(warning.color);
    document.getElementById("foregroundRemaining").classList.add(alert.color);
    document.getElementById("textId").classList.remove('textPop');
    document.getElementById("textId").classList.add('textPop2');
  }
  else if (timeLeft <= warning.threshold)
  {
    document.getElementById("foregroundRemaining").classList.remove(initial.color);
    document.getElementById("foregroundRemaining").classList.add(warning.color);
    document.getElementById("textId").classList.remove('text');
    document.getElementById("textId").classList.add('textPop');
  }
}

//calculate time fraction for circle
function calculateTimeFraction() {
  const rawTimeFraction = timeLeft / TIME_LIMIT;
  return rawTimeFraction - (1 / TIME_LIMIT) * (1 - rawTimeFraction);
```

40210913 David Montgomery Individual Contribution

Dog fight mini game

Javascript for moveDog( ); I wanted to create an engaging and immersive gameplay for the dog fight level of iSPY. I used a HTML button to represent the dog and allow the player to click (attack) it.



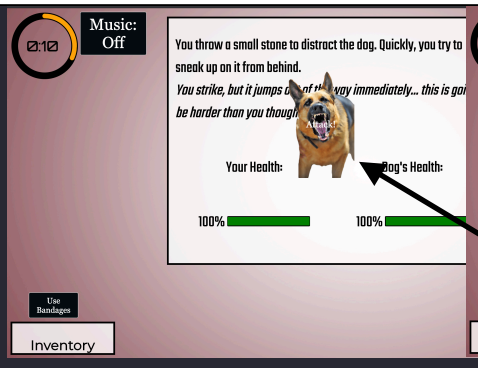
Javascript

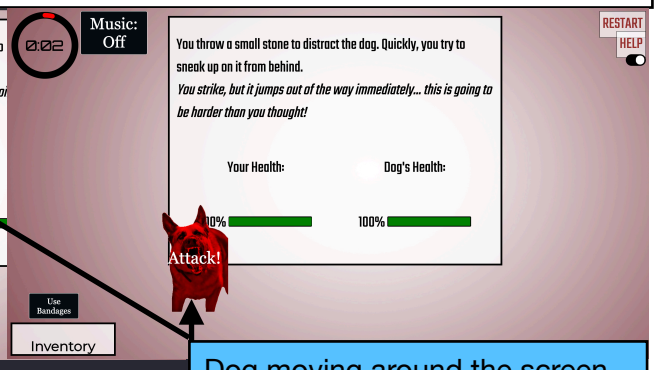
```
// method to move dog button randomly around the screen, used onhover
function moveDog() {
  var dog = document.getElementById('attack');
  var x = Math.floor(Math.random()*97);
  var y = Math.floor(Math.random()*97);
  dog.style.top = x + '%';
  dog.style.left = y + '%';
  dog.style.transition = '1s';  //code to adjust speed that dog moves away
}
function inflictDamageToDog() {
  document.getElementById('lastAttack').style.display = 'inline-block';
  damageNumber = Math.floor(Math.random() * 40) + 20;

  if ((dogHealth - damageNumber) <= 0) {
    document.getElementById('attackSoundEffect').play();
    damageDealt += damageNumber;
    localStorage.setItem('damageDealt', damageDealt);
    localStorage.setItem('damageTaken', damageTaken);
    localStorage.setItem('causeOfDeath', causeOfDeath);

    dogHealthValue.style.width = 0;
    dogHealthLabel.innerHTML = "DEAD";
    document.getElementById('dogIntro').style.display = 'none';
    document.getElementById('attack').style.display = 'none';
    document.getElementById('lastAttack').style.display = 'none';
    document.getElementById('wonFight').style.display = 'block';

    timeDisappear();
    stopTimer();

  } else {
    document.getElementById('attackSoundEffect').play();
    dogHealth -= damageNumber;
    damageDealt += damageNumber;

    dogHealthValue.style.width =  dogHealth + "%";
    dogHealthLabel.innerHTML = dogHealth + "%";
    recipient.innerHTML = "The Dog";
    lastAttackDamage.innerHTML = damageNumber + " points of";
    setTimeout(inflictDamageToPlayer, 2000);
  }
}
```

I then used CSS with a transition effect on hover of the dog image/button, to show the player that they have lined up an attack and are ready to click. This animation changed the background image of the dog to red, and enlarged the 'Attack!' text. Javascript was then used to adjust the CSS, moving the dog randomly when you get near it, as if it was jumping out of the way of your attacks. Javascript was further used on click (attack) of the dog, to take health from the dog, make a sound effect, and eventually kill the dog and display a 'MISSION ACCOMPLISHED' div to take you to the next level, or timeout and display a 'MISSION FAILED' div to take you to the stats page.

CSS

```
#attack {
    position: absolute;
    border: none;
    background-color: #050a14;
    color: white;
    text-align: center;
    font-size: 28px;
    padding: 20px;
    width: 200px;
    height: 321px;
    -webkit-transition: all 0.2s;
    -o-transition: all 0.2s;
    transition: all 0.2s;
    cursor: pointer;
    margin: auto;
    background: url('images/dog3.gif');
    padding: 0;
  }
```






```
#attack:hover {
  border: none;
  background: url('images/dog3hit.gif');
  font-size: 50px;
}
```

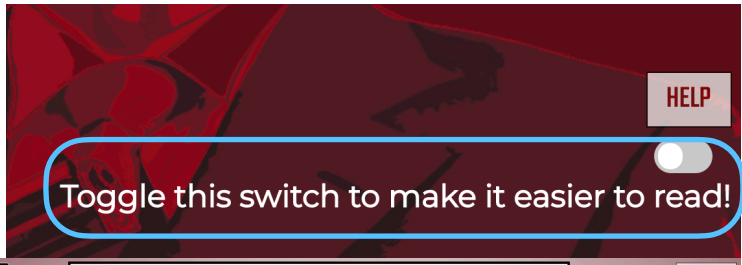
Dog moving around the screen randomly 'onhover' and 'onclick'

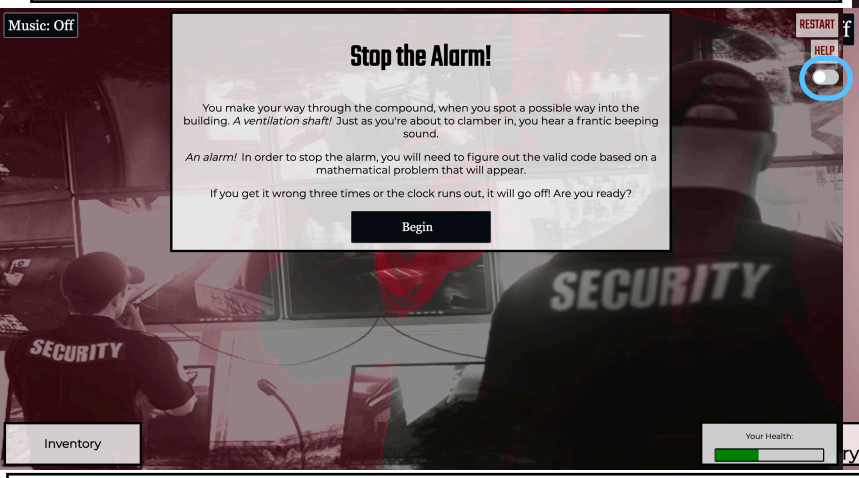
Also an example of 'easy read' mode being toggled.

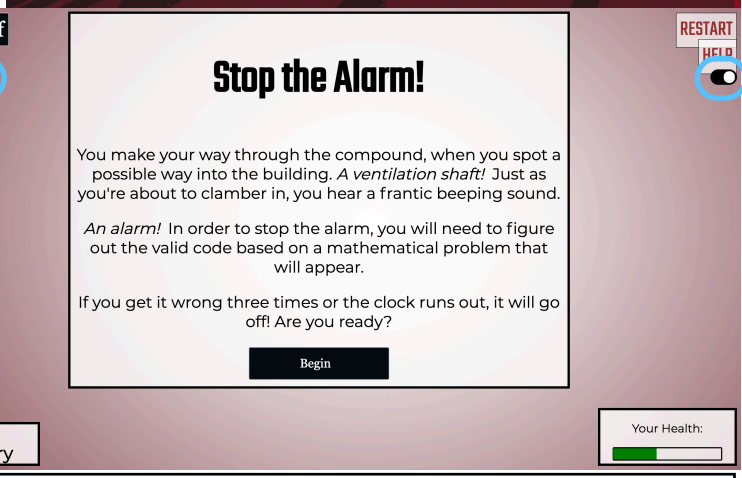
40210913 David Montgomery Individual Contribution

Accessibility feature: ‘Easy Read’ Mode

I implemented an easy-read feature to all pages and adjusted all CSS to respond to the greater font size and still display and function correctly.
This was a complex feature to implement at the end given the differing styles of each person’s code. I had to check







the HTML & CSS tags, test the effects of the increased font size, and then write the Javascript to adjust the CSS to ensure that everything still displayed correctly. I then had to ensure that it worked similarly and effectively on each page. I built the toggle switch using HTML and styled it to suit our theme, using CSS and an animation to show the user which state it was in. I then used Javascript so that when the switch was toggled, it would enlarge the font size, change the background to a plain gradient, and then adjust the sizes of containers to accommodate these changes.

CSS

```css
/* The switch - the box around the slide */
.switch {
  position: absolute;
  top: 140px;
  right: 10px;
  display: inline-block;
  width: 60px;
  height: 34px;
}

/* Hide default HTML checkbox */
.switch input {
  opacity: 0;
  width: 0;
  height: 0;
}
```

CSS

```css
.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #ccc;
  -webkit-transition: .4s;
  transition: .4s;
}

.slider:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  -webkit-transition: .4s;
  transition: .4s;
}
```

CSS

```css
input:checked + .slider {
  background-color: black;
}

input:focus + .slider {
  box-shadow: 0 0 1px black;
}

input:checked + .slider:before {
  -webkit-transform: translateX(26px);
  -ms-transform: translateX(26px);
  transform: translateX(26px);
}

/* Rounded sliders */
.slider.round {
  border-radius: 34px;
}

.slider.round:before {
  border-radius: 50%;
}
```

HTML

```html
<label class="switch">
  <input type="checkbox" id="easyReadCheckbox" onclick="easyRead();">
  <span class="slider round"></span>
</label>
```

When checkbox/toggle switch = checked, easyread( ) method is called.
When switch = unchecked (clicked again), page is set back to normal.

Javascript

```javascript
function easyRead() {
 // Get the checkbox
  var checkBox = document.getElementById("easyReadCheckbox");

  // If the checkbox is checked, display the output text
  if (checkBox.checked == true){
    document.body.style.backgroundImage = "url('../whiteBackgroundImage/whitebackground.jpg')";
    document.body.style.fontSize = '150%';
    document.getElementById("centreBox").style.marginTop = "-15%";
  } else {
    document.body.style.backgroundImage = null;
    document.body.style.backgroundImage = "url('images/dog.jpg')";
    document.body.style.fontSize = '100%';
    document.getElementById("centreBox").style.marginTop = "-10%";
```