CS230 - Web Information Processing

Assignment 2

Assignment Release Date:	21-02-2022
Submission Due Date:	04-03-2022
Feedback Due Date (estimated):	14-03-2021 (for assignments that make Due Date)
Support Laboratories	Lab 03, Lab 04 (Two Weeks)
Total Mark:	10%

This Assignment is worth 10% of the Web Information Processing CA Component.

This is an open-book, graded assignment. You may use online resources for reference purposes only to help with the assignment. Please cite all references as comments in your submissions. You cannot directly reuse HTML/CSS/JS **solution code** from online sources. **You must not engage with another student, in person or electronically (phone, social media, etc.) to secure assistance with this assignment. If you do so you will receive an automatic fail (0%).** We will perform similarity checks on submitted assignments to check for collaborative efforts. A reasonable attempt at this assignment will gain you 10% of your continual assignment marks.

Assignment 02 - Support Vector Graphics

SVG is to graphics what HTML is to text. It is a text-based (XML), open Web standard for defining vector-based images that can be rendered cleanly (scaled/zoomed without loss of quality) at any resolution, and are designed specifically to work well with other web standards such as DOM, CSS, JavaScript, XSL, etc. SVG has been developed by the World Wide Web Consortium (W3C) since 1999. SVG 1.1 became a W3C Recommendation on 16 August 2011.

SVG images and their accompanying behaviours are defined using pure XML text files, which means they can be searched, indexed, scripted, etc. Every element and attribute of the image can be animated. They can be created and edited with any text editor, with drawing software, or created using programs (like we will be doing in this module).

Compared to classic (bitmapped) image formats such as JPEG or PNG, vector images such as SVG can be rendered at any resolution without loss of quality. They can be easily localised, without the need of a graphical editor, by updating the embedded text using a text editor. With a proper understanding of their construction, and using libraries or custom-written programs, SVG files can be localised interactively.

Assignment 02 - Requirements

You are required to create an SVG “Ghost Animator” as a (HTML/CSS/JS) web app. You may use the default Ghost SVG image provided (See Figure 1, below) or create your own Ghost image. Your app should implement a user interface that facilitates the creation of an animation sequence that is automatically embedded in the Ghost SVG as an animation. You app should provide controls for implementing SIX different animation effects, together with (i) a button that plays the animation when pressed, (ii) a button that toggles between the SVG image and the Image source, and (iii) a

button that exports (downloads) the SVG. The downloaded SVG must be an animated ghost image that runs as an animation when loaded in a browser or SVG viewer.

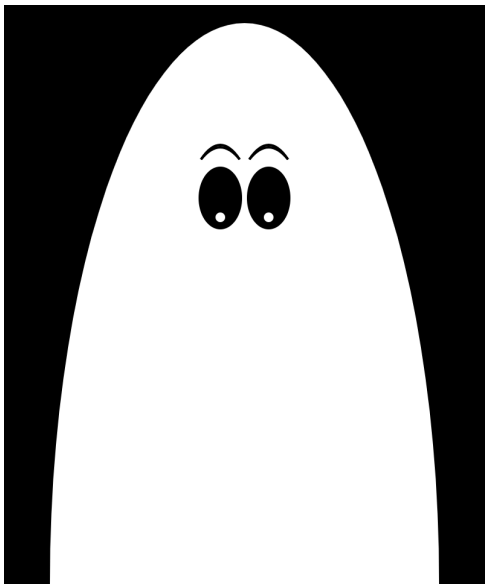

Figure 1
Sample Ghost SVG Image

There are no specific requirements for the layout of the controls for animating elements. You need to include SIX different animation effects that are editable using User Interface (UI) controls. For example, you could implement controls for animating the size and position of the eyes, brows, eyeballs, eye shape, etc. You could also implement moving the Ghost body, for example, zooming, panning, rotating, or a combination of these effects.

Animations have a starting and ending state. The transition from the starting to ending state can be short, for example a few seconds, but it must be long enough that the animations can be observed by the viewer. You need to decide appropriately on the duration of the animation in addition to the effects.

Please note that moving a groups of items simultaneously is considered a single animation, i.e, moving two eyes, two brows, two eyeballs moves six facial elements, but it is a single animation.

Assignment 02 - Development Notes

Please adhere to the following development requirements:

1. All of the Ghost AnimatorUser Interface elements must be generated using HTML elements together with appropriate CSS styling and layout.
2. You may **not** use an SVG framework, or SVG JS library, for this assignment. For example, you may not use D3.js, Snap.svg, SVG.js, etc. All code must be developed by you.
3. You may **not** use a CSS framework, such as Bootstrap, for this assignment. You may, if you wish, use the jQuery Javascript framework. If you use TypeScript, or similar, and transpire to JavaScript,

you need to provide all sources. Your app only needs to run on a desktop browser such as Chrome - you do not need to ensure it works on every browser (mobile browsers, for example).

4. You **must** comment your code, clearly indicating, how your code implements the Ghost Animator described above in the "Assignment 02 - Requirements" section. You will lose marks for uncommented code.

5. A Sample UI structure is given in Figure 2, below. There should be three main zones; one for the Animation Controls, one for the Buttons, and one for the SVG or Source Code Display. You may style using your own preferences. The example shows is a minimal expectation for the UI.

6. While it is fine to consult online SVG Animators, and accompanying articles, for references, you may not re-use code from these projects. Please cite your reference sources in your codebase. You may use any of the code provided by John Keating in his EU Code Week 2021 Seminar (https://github.com/JKeatingMU/EUCodeWeek2021) Please indicate clearly if you have used any of this code. John Keating's EU Code Week SVG Tutorial can be found here: https://youtu.be/_jumZw12Jn4.

7. The SVG code for the Sample SVG Ghost Image shown in Figure 1 is given in Figure 3, below. You may choose to use, or modify, this as your base (starting) Ghost SVG. There are other examples that you can use in John Keating's EU Code Week 2021 Seminar (https://github.com/JKeatingMU/EUCodeWeek2021). Or you can create your own Ghost SVG.

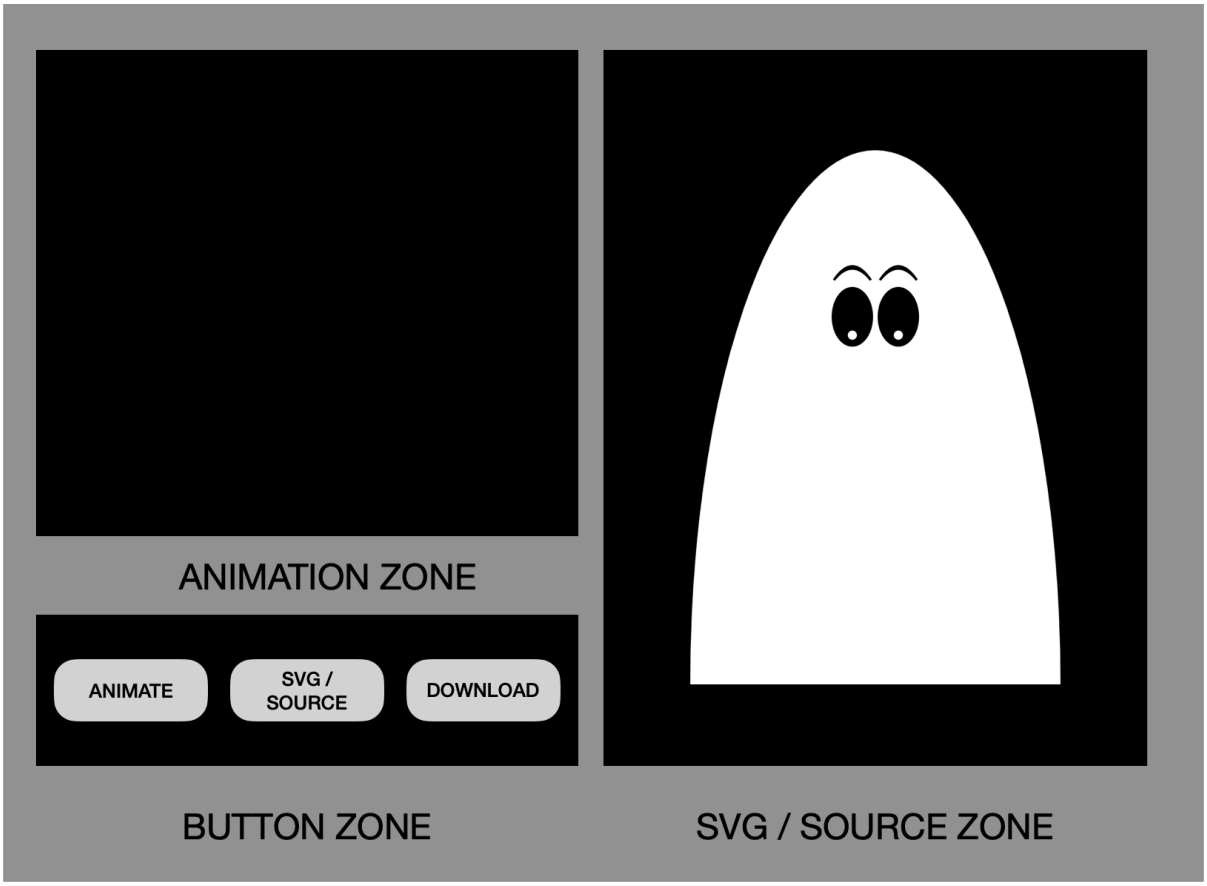

Figure 2
Sample Minimal UI Layout

```
<svg width="400" height="480" viewBox="0 0 100 120">
    <rect id="background" width="100" height="120" style="fill:black; stroke-width:1; stroke:black"/>
  <g id="ghost" transform="rotate(0), translate(0,0), scale (1.0)">
     <path id="body" d="M 10 120 A 2.0 5.8, 0 0 1, 90 120" style="stroke:white; stroke-width:0.5; fill:white"/>
    <g id="eyes" transform="rotate(0,50,40), translate(0,0)">
      <g id="eye-l" transform="rotate(0,50,40), translate(0,0)">
            <ellipse id="eye-l-ball" cx="45" cy="40" rx="4" ry="6" style="stroke:black; fill:black;"/>
        <rect transform="rotate(16,46,35)" x="40" y="33" width="11" height="6" fill="white" visibility="hidden" />
        <circle id="eye-l-pupil" cx="45" cy="44" r="0.5" style="stroke:white; fill:white"/>
        <g id="eye-l-brow">
                    <path d="M 41 32 Q 45 27 49 32" stroke="black" stroke-width="0.5" fill="transparent"/>
                    <path d="M 41 32 Q 45 26 49 32" stroke="black" stroke-width="0.5" fill="transparent"/>
        </g>
      </g>
      <g id="eye-r" transform="rotate(0,50,40), translate(0,0)">
            <ellipse id="eye-r-ball" cx="55" cy="40" rx="4" ry="6" style="stroke:black; fill:black"/>
                <rect transform="rotate(-16,56,35)" x="49" y="32" width="11" height="6" fill="white" visibility="hidden" />
        <circle id="eye-r-pupil" cx="55" cy="44" r="0.5" style="stroke:white; fill:white"/>
      </g>
      <g id="eye-l-brow">
                  <path d="M 51 32 Q 55 27 59 32" stroke="black" stroke-width="0.5" fill="transparent"/>
                  <path d="M 51 32 Q 55 26 59 32" stroke="black" stroke-width="0.5" fill="transparent"/>
      </g>
    </g>
  </g>
</svg>
```

Figure 3
Sample Ghost SVG Image - SVG Code

IMPORTANT SUBMISSION DETAILS

Before submitting your assignment students should check that their solution works in Chrome and/or Firefox. Please indicate the Browser, Lab Operating System (Linux/Windows) and Browser version used for testing (as a comment in your submitted code).

All work must be submitted via Moodle (see "Assignments" section for submission). Work submitted via other means will not be accepted unless you have prior arrangements with the Head Demonstrator (Behnam Faghih). All work MUST be submitted by the due-date deadline. Late submissions will not be accepted.

*The assignment submission is a zip file named "**assignment-02-xxxxxxxxx.zip**" (where "xxxxxxxxx" is your student id) containing a solution file named "**assignment-02.html**" together with any other resources used in the assignment solution. External CSS and Javascript files should be named "**assignment-02.css**" and "**assignment-02.js**", respectively. Please ensure that all external files use relative directory referencing, rather than hard-coding the files' location.*