Computer Science Tripos

Part II Project Proposal Coversheet

Please fill in Part 1 of this form and attach it to the front of your Project Proposal.

Part 1

Name: Radzim Sendyka

CRSID: rs2071

College: Churchill

Overseers: (Initials) rkm38, amp12

Title of Project: Photosensitivity Safety

Date of submission: 15.10.2021

Will Human Participants be used? No

Project Originator: Myself

Signature: ----------

Project Supervisor: Andrea Ferlini

Signature: ----------

Directors of Studies: John Fawcett

Signature: ----------

Special Resource Sponsor:

Signature: ----------

Special Resource Sponsor:

Signature: ----------

Above signatures to be obtained by the Student

Part 2

Overseer Signature 1: ----------

Overseer Signature 2: ----------

Overseers signatures to be obtained by Student Administration.

Overseers Notes:

Part 3

SA Date Received:

SA Signature Approved:

Introduction

The goal of the proposed project is to create a system that will assist people with photosensitive epilepsy navigate the world wide web safely.

Photosensitive epilepsy is a type of epilepsy that can be triggered by flashing lights. The exact estimations of the number of people who suffer from it vary, but all are in the millions. Like every type of epilepsy it is extremely dangerous, as each attack can be fatal due to injuries sustained during the incident, and every attack causes brain damage that can be permanent.

With the advent of film, people have realised that TV and other bright screens that display video content have the potential to cause seizures in people suffering from photosensitive epilepsy. Most countries developed legal countermeasures to protect people from this content. In the UK, for instance, the Ofcom Broadcasting Code states that "Television broadcasters must take precautions to maintain a low level of risk to viewers who have photosensitive epilepsy". However, no such regulations exist for the world wide web, which can also contain triggering content that is not being monitored. Furthermore there exist numerous reports of such content being deliberately sent to people with the intent to cause them physical harm. What is more, and difficult to understand, such malicious acts are not illegal under UK law. These circumstances make a client-side automated content filter crucial for people with photosensitive epilepsy to be safe online.

While it is not possible to perfectly identify all triggers in online content, one can apply guidelines pertaining to TV and film to the content being displayed on personal computer screens. There are very specific instructions laid out by the W3C that describe the maximum levels of flashing content that is acceptable, and it is possible to implement a program that analyses videos to see if they pass the criteria.

I would like to create a system that an end user could use to shield themselves from exposure to triggering content, both malicious and accidental. I decided that a web browser extension would be best suited for this task. The extension would analyse all videos embedded in the site (and potentially other dynamic elements) or possibly the entire display, looking for specific patterns of blinks and flashes that occur at the frequency, brightness and size which could make them a trigger, and warn users of potential dangers.

Project structure

Web Browser Extension

- I have chosen a web browser extension as the most appropriate way of screening for dangerous content. This is due to the ease of use and installation for the end user, as well as the fact that it provides straightforward access for the system to analyse the web content being displayed.
- The extension will scrape the website content for videos and possibly other dynamic elements. These will be passed to an analysis engine. Analysing rendered website content is a possible extension.
- The extension will take the results of the analysis and based on the verdict either declare the video safe (W3C compliant), or unsafe, and modify the displayed website to reflect that information, along with relevant details.
- The extension will use JavaScript code. Communicating with the analysis engine (running in Python) will not be easy, as Python is not generally supported with JavaScript, and may require major changes to the project plan - reworking the engine code into JavaScript, either manually or using a transpiler, or implementing a server where the Python code can be run.

Video Analysis Engine

- The main part of the Engine part of the project will be implementing the guidelines, and efficiently analysing the videos to screen for any parts that do not meet the criteria.
- Those criteria are available in the W3C: SC 2.3.1 (www.w3.org)
    - *"Three Flashes or Below Threshold: Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds."*
    - Where "Flash" is defined as *"a pair of opposing changes in relative luminance that can cause seizures in some people if it is large enough and in the right frequency range".*
    - Specific definitions of general flashes, red flashes, frequency ranges, luminance, viewing size, and exceptions are also provided in the W3C: SC 2.3.1 document, and will be implemented in the system.
    - "There can be no more than three general flashes and/or no more than three red flashes within any one-second period" [www.w3.org]. This needs to be tracked, and evaluated live, with as little delay as possible - possibly displaying a warning when a second flash occurs in cases where the next 1 second of video has not yet been analysed.
    - "The combined area of flashes occurring concurrently occupies no more than a total of .006 steradians within any 10 degree visual field on the at typical viewing distance - using a 341 x 256 pixel rectangle anywhere on the displayed screen area when the content is viewed at 1024 x 768 pixels will provide a good estimate of a 10 degree visual field for standard screen sizes and viewing distances" [www.w3.org]. This poses a challenge, as a 1024x768 pixel matrix contains as many as 349696 possible distinct 341x256 rectangles. Evaluating all of them would be impossible to do in real time, and therefore various search and exclusion techniques have to be devised.
    - "A general flash is defined as a pair of opposing changes in relative luminance of 10% or more of the maximum relative luminance where the relative luminance of the darker image is below 0.80" [www.w3.org] - in digital terms, the relative luminance of a colour is defined as L = 0.2126 * R + 0.7152 * G + 0.0722 * B where R, G and B are defined as: if RsRGB <= 0.03928 then R = RsRGB/12.92

else R = ((RsRGB+0.055)/1.055) ^ 2.4, and RsRGB = R8bit/255 (with analogous calculations for G and B) [www.w3.org].
    - Tracking transitions - their relative luminance differences, transition directions, and locations, in order to pair them with others and record them as flashes (in very low computational complexity) will be a crucial and challenging task.
    - A red flash is defined as any pair of opposing transitions involving a saturated red." - in digital terms: "for either or both states involved in each transition R/(R+ G + B) >= 0.8, and the change in the value of (R-G-B)x320 is > 20" [www.w3.org].
    - "Exception: Flashing that is a fine, balanced, pattern such as white noise or an alternating checkerboard pattern with "squares" smaller than 0.1 degree (of visual field at typical viewing distance) on a side does not violate the thresholds." [www.w3.org] - this allows for a faster analysis that does not need to consider every pixel individually, but considers only the average luminance and colour values of smaller squares, as fine, complementing flickers inside squares are exempt.
- The analysis should be computationally reasonable, and happen in quasi real time, with minimal pre-processing delay.
- Videos will be analysed using python. Computations will be done on a timeseries of frames, using NumPy matrix operations. Individual frames will be transformed to calculate the sRGB values, relative luminance, red saturation, red colour majority, and other crucial criteria for each pixel. These values will be compared with neighbouring frames to detect transitions. These transitions will be analysed to detect and classify potential flashes.
- Creating this analysis engine will involve knowledge about human perception, colour spaces, visual fields, and understanding of the conversions between physical and digital units. I acquired during the Introduction to Graphics and Further Graphics courses.
- I acquired other crucial skills, such as familiarity with Python, most notably NumPy matrix computations during the Data Science course, and some personal projects.
- This will be using the CPU. The impact of the plugin on the CPU load will be evaluated to ensure it's suitable for end users. Running calculations using the GPU is a possible extension.

Data structures, algorithms, and evaluation:

- Analysing videos means working with that kind of data. Data structures will include those for video storage, and a timestamp-indexed array of events for the analysed videos.
- Lookup will use a hash table or similar low complexity class solution for hash value, filename and address comparisons.
- Test data will include captures of typical internet content – popular videos, and screen captures of popular websites that contain dynamic elements. Content will be evaluated with PEAT and compared to the results obtained by the system. Due to the limitations of PEAT (in terms of video format, codec, resolution, etc.), test data will have to be created specifically for this purpose.

Extensions:

- It would be extremely inefficient to re-analyse the same content every time a page is refreshed or accessed again. In order to combat this, the system will include a way of remembering the results of all analysed videos using their metadata and hash value as keys. This opens up the possibility of a centralised system that gathers lookup table data from everyone (subject to an opt-in setting), and shares it with others, allowing for even greater reductions to the computational loads. Data gathered this way could be an extremely

interesting insight into the prevalence of photosensitivity triggering content in various websites across the internet, as well as some general patterns present.

- Allowing user input data, such as personalised frequency sensitivity, visual field estimations, to conduct analyses more suited to individual needs.
- Using wavelets or other techniques to detect movement artifacts - differences between frames that do not result from flashing, but from ordinary movement, yet are hard to exclude from the analysis.
- Analysing the entire screen while browsing, and tracking the risk to the user, with possible warnings.
- Improving quasi-real time behaviour by using the GPU could be a possible extension, especially if CPU-solutions underperform.
- A possible extension to the project is masking the detected flashes by modifying content, so that the returned video can be safely viewed. This extension would be very hard and require much time and effort.

Success criteria

The project will be a success if a working web browser extension is created, that screens web content for flashing images, using an engine that implements the W3C guidelines.

The analysis engine will be a success if it is able to screen for triggers quickly enough for working with a web browser, and well enough that it approximates the results achieved by the lengthy and expensive computations done by the standard tools used in industry (PEAT and/or Harding FPA – Harding FPA is a tool meant for use in TV and film, and follows different guidelines than those set by W3C, so differing results are to be expected).

Plan of work

General milestones

- November 1st 2021 - Deciding on the project configuration elements such as IDE, version control, and browser. Basic "proof of concept" video analysis with a basic blueprint of the functionality.
- November 15th 2021 - Creating a mock browser extension that contains a basic blueprint of the functionality.
- January 18th 2022 - Creating the engine, connecting it with the extension, so that it becomes a working prototype.
- February 18th 2022 - Further enhancing to reach a state where the project is fully developed.
- March 18th 2022 - Improving the system based on own testing and feedback from supervisors.
- 31st March 2022 - Finalising the system.
- 1st April 2022 - Starting the dissertation text based on the project.
- 1st May 2022 - Finalising the dissertation text, with charts, figures, examples, and appropriate source code fragments.
- Dissertation and source code submission by Friday 13 May 2022.

The workload is heavier at the beginning of the project due to uneven course load.

Browser extension component milestones

- First step of the project will be creating a blank browser extension, with some sample functionality. This should be achieved in November 2022.
- Then coding that extension to detect videos (and possibly other dynamic elements) in the website, so that they can be analysed. This should be achieved in November 2022.
- Implementing a way of checking if the video is already present in the lookup table, and if the video is not present there, passing that video to be analysed by the engine. This should be achieved in January 2022.
- Add a way of displaying the results of the analysis on the website. This should be achieved in January 2022.

Engine milestones

- Specifying an implementation of the W3C guidelines that will be as close to while also computationally realistic for the purpose of a browser extension. (This sub-task will be hard and require much time and effort). This should be achieved in October 2021.
- Writing pseudocode, deciding on the language and tools most appropriate for the task at hand. This should be achieved in October 2021.
- Implementing the engine (This sub-task will be very hard and require much time and effort). This should be achieved in December 2021.
- Creating a way of noting the results down in a format that can be shared with the lookup table and returned to the extension. This should be achieved in January 2022.
- Returning the results to the browser extension. This should be achieved in January 2022.
- Comparing results with those returned by PEAT/HFPA on the same video samples. This should be achieved in February 2022.
- Complexity and performance analysis, including CPU load and power consumption. This should be achieved in February 2022.

Writing up

- Documenting the process will be done during the entirety of the project. This includes code that is written and manual notes on the progress of the project.
- Gathering results will be done during the entirety of the project.
- Writing the dissertation will start around the end of the system's finalisation, (March 2022).
- Dissertation and source code submission by Friday 13 May 2022.

Starting point statement

I have not written any significant bodies of code or other material for this project before the start of the 2021/2022 Academic Year.

To the best of my recollection I have never created web browser extensions, or video analysis tools, and do not intend to reuse any such code.

Resource declaration

My project will be developed using my own machine (personal laptop). *I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.* These contingency plans include access to another machine, automatic cloud backup (Microsoft OneDrive), and periodic physical backups on an external hard drive.

My project will involve a commonly used browser as it is aimed to protect internet users. The aim is to use a browser that is available to the largest number of users, so Google Chrome is preferred, as it is the most widely used browser. Alternatively Mozilla Firefox may be used.

My project will likely involve existing photosensitive epilepsy analysis tool: the PEAT (free), and possibly Harding FPA or another commercial tool. They will be used for ground-truth approximation, and testing.