Project: Proposal

Assessment 1:

Mitchell Bellamy (BEL13374080)

Project (CMP3060M)

10/16/15

Contents

1. Introduction 2
1.1. Project Domain and Rationale 2
1.2. OBS Plugin to overlay key presses 2
1.2. Research Methods 4
1.3. Tools 4
2. Aims and Objectives 5
2.1. Aim 5
2.2. Objectives 5
3. Literature Review 7
3.1. Learning and Memory association 7
3.2. Market and Existing solutions 8
3.3. Summary 10
4. Project Plan 11
4.1. Development Life Cycle 11
4.2. Development Methodology 11
4.2. Gantt Chart 12
4.3. Gantt Chart Milestones 13
4.4. Risk Matrix 14
5. References 16

1. Introduction

1.1. Project Domain and Rationale

This project involves elements of software development, open source integration, human-computer interaction and design. Many of these aspects will be tackled in some way throughout the introduction and explained as to why they are important areas for the project and what will be done to fulfil them.

It is these aspects and the ability to produce a deliverable to users across the globe that made this project desirable. On top of that, it provides good grounds to gain experience of working in the open source sector of industry which can improve employability and software knowledge.

This project outlines the requirement to develop a plugin (software) for Open Broadcaster Software (OBS). OBS is a program that allows the user to stream a video feed live to the internet (through services such as Twitch.tv and Livestream.com). This plugin is to display relevant keystroke information to viewers of the stream.

1.2. OBS Plugin to overlay key presses

Open Broadcaster Software (OBS) is an open source program developed for people to record and/or live stream video from their computer to services such as Twitch.tv and livestream.com. The use case scenarios are vast for live streaming, such as festivals, talk shows, games, podcasts and more. In particular, live streaming from home on a computer has become exceptionally popular with consumers and content creators over the last couple of years (Twitch.tv, 2013; YouTube, 2015). This high growth and uncertain future makes the streaming community a very interesting area for new research and development.

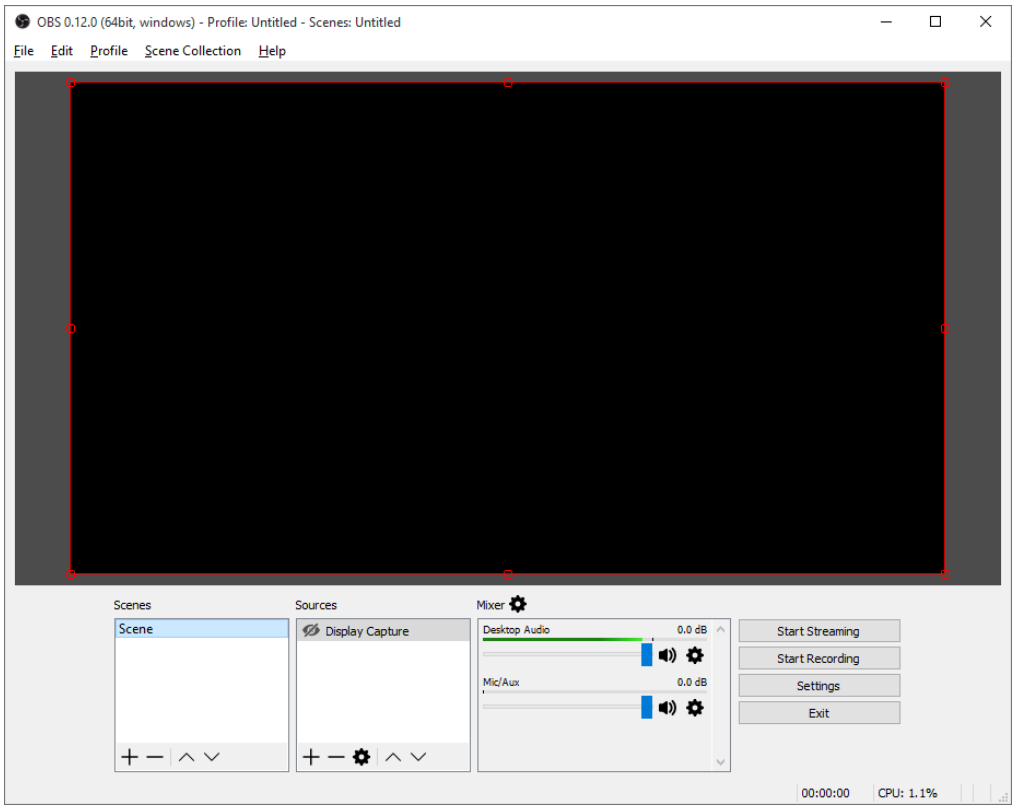

Figure 1: OBS Studio 0.12 on Windows.

OBS Studio allows the user to both stream and record video. More uniquely, is that as a streaming software package, it offers ‘scenes’ and ‘sources’ which can help create independent windows to

help separate what the streamer sees on their computer to what the viewers see online. This is something that will need to be considered and incorporated to help empathise with the viewers and users independently.

Even now ‘live streaming’ video games in particular can be a very heated topic. There are many who believe ‘e-sports’ are now mainstream or soon to be widely accepted (Twitch.tv, 2013). The community surrounding just e-sports are extremely passionate and consistently consume this media. One of the more interesting growth areas for this project to look into however is those that watch professional players to learn how to play better for themselves. Furthermore, for those that record tutorial videos with OBS for computer software for example, have very limited options in displaying shortcuts or key presses to help viewers learn more efficiently. These are just two scenarios that would benefit from a plugin to overlay key presses during live streaming and recording.

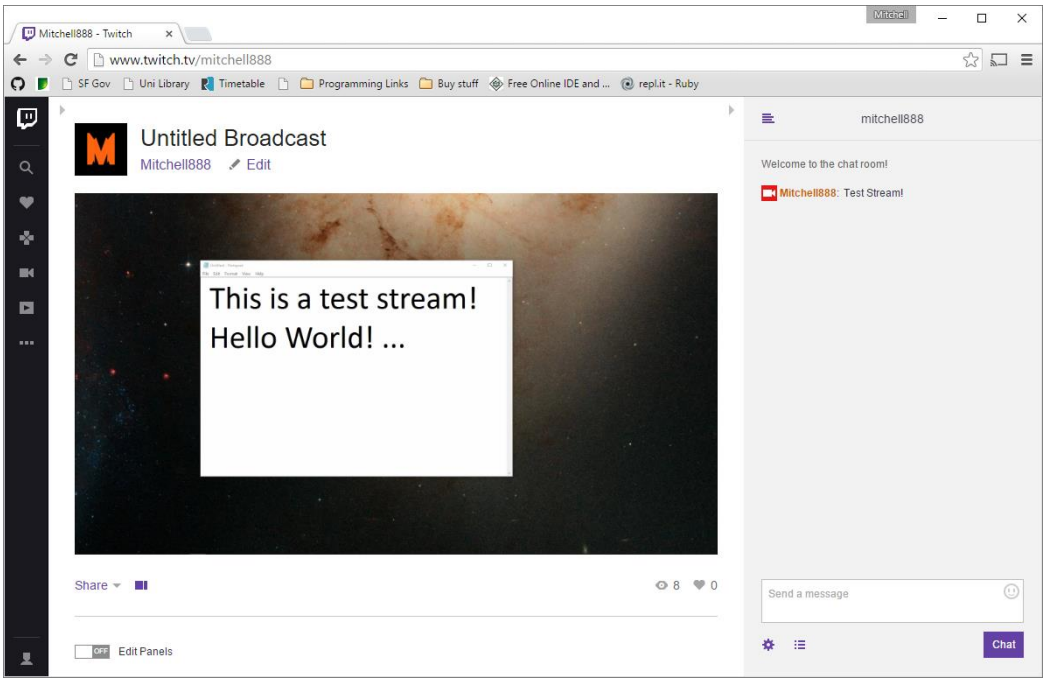

Figure 2: A quick test stream on Twitch.tv

Twitch.tv is just one of the many services that offer to host live streaming, along with YouTube and Livestream.com. Twitch.tv however has a primary focus on gaming and related content (such as music, software and talk shows). YouTube also has a live streaming sub-site which focuses on gaming content only, whereas Livestream.com aims to host live events.

For this project, Livestream.com is somewhat out of scope as it is unlikely that live events would benefit from a key overlay. Therefore, the aim of this project will target services similar to and including Twitch.tv. This helps to define a domain and scope for the aim, which will help to create potential focus groups or for testing purposes.

As for the plugin itself which is to be developed in C++ on the latest version of OBS Studio, will be required to effectively display relevant keystrokes to the viewer but not overlay this information on the streamers machine as to avoid distraction and clutter. The streamer will always be able to see what the viewers see through OBS Studio itself.

The design of the key overlay needs to display information in such a way that is helpful to viewers so that they can follow the streamers actions (and reactions) in a way that has not been possible

before. The display of this information is something that will need to be researched, trialled and tested to ensure it delivers the aim to the viewers.

1.2. Research Methods

To effectively evaluate the outcome of this project, the artefact must be researched against. Due to the software engineering nature of this particular project, it would be more beneficial to do both research and gather feedback on the success of the artefact.

Feedback can come from several places, some observational ethnography may work in both research and evaluative purposes for gathering information and feedback in assessing the plugin. Other programs and software will also be researched, both in terms of design and implementation in order to see what works well and why. It may also prove to be a good idea to present this plugin (when it is at a suitable stage in development) to the open source community surrounding OBS studio. Gathering their opinions and feedback to fuel further design and implementation aspects will clearly show how effective the artefact is during evaluation.

1.3. Tools

Many of the tools associated with this project are aimed at assisting development in an effective way. Especially for peace of mind and security, GitHub and OneDrive will both be used as to not place sole reliance on any one system.

GitHub (Github Inc, 2015) is a service based on git for version control. Version control is extremely useful for Open Source developments, documentation and sharing of code. Having a repository on GitHub will make the final plugin much more accessible to the Open Source community surrounding OBS studio. GitHub also provides the facility to write wiki style pages (such as user guides and documentation).

OneDrive (Microsoft, 2015) is a cloud-based web storage service for backups and access across devices. This complements version control services such as GitHub as OneDrive will automatically upload files to its servers so if anything were to happen to GitHubs repository or the personal computer the plugin is being developed on, then the files will be easily accessible on OneDrive. This mitigates risk of loss of data.

Visual Studio is an Integrated Development Environment for many languages, systems and devices. As OBS Studio is programmed in C++ (with specific coding styles) it is also recommended that Visual Studio is used alongside CMake for Windows users to provide cross platform compatibility.

2. Aims and Objectives

2.1. Aim

The aim of this project is to develop a plugin which enables the user to display their corresponding keystrokes on a video recording (including live streaming) via Open Broadcaster Software (OBS).

Target Audience:

- For e-sports enthusiasts, gamers and spectators.
- For viewers of software based tutorials.

2.2. Objectives

- Conduct further artefact research

In order to correctly identify critical features of the plugin, it is essential to perform some elements of Human-computer studies to evaluate how the plugin should be designed. Amongst this, it should also influence the graphical interface of the plugin to ensure excellent usability. This research may include; Observations, Surveys, Code reviews and evaluations of other implementations.

- Setup Development Environment and build

To begin development, the environment and dependencies need to be in place according to the standards set by the community for OBS Studio. Within this objective is the requirement to setup CMake (A solution building tool for cross-platform development) for cross-platform support.

It should be noted that there are also coding standards for C++ that OBS Studio is written in and therefore should be followed closely to ensure it can be easily documented and understood.

- Implement key features

Some requirements gathering will occur naturally during further research, observations and surveys. However, there are some aspects of this project that must be met. These are;

i. Key Presses are displayed to the viewer in some meaningful way.
ii. Do not store system wide key presses.
iii. Contributes to the Open Source community.
iv. Must interact with OBS Studio.

These requirements are somewhat flexible, but it is these aspects that provide the scope and domain for the plugin development.

- User study and Testing

To help understand how the plugin would be used in a real scenario, a User study could provide excellent feedback to critique the plugin and provide insight into what could be improved in future iterations. Alongside this stage, testing can occur to ensure the plugin behaves as expected and adheres to the coding standards set by the community of OBS Studio.

- Ensure plugin behaves ethically

The plugin must not *collect* keystrokes. This behaviour is typically reserved for ‘key loggers’ that attempt to store sensitive information by storing every key press. It is in the Users best interest that this behaviour is avoided in the plugin.

- Project Report

The final report will be documenting the entire process, including discussions made in this document. The research methods and tools will be evaluated. The objectives and aims critically discussed to discover just how the artefact and project turned out. Both the literature and later research can be compared to formulate an idea of whether the artefacts design and implementation were successful in relation to the aim. All of these discussions will need to take place as development progresses and this will be shown in the plan at a later section.

3. Literature Review

3.1. Learning and Memory association

To support the aim of this project, research into areas of learning may guide software development and design to improve the plugins ability to help viewers memorise and implement key presses they see online. This area of research is particularly interesting as the viewers of such videos and live streams are unable to memorise these key presses through physical movement at the time of viewing.

In the Journal article 'Parallel neural networks for learning sequential procedures' (Hikosaka et al, 1999, 465) a concept of procedural learning (learning how to perform new tasks) is proposed which does not 'strictly' rely on sensorimotor processes performed in sequence. Thus, it can be said that a viewer of a particular stream may be able to learn key presses through memorisation/recognition at the early stages of learning new movements but then form the association with motor control, its corresponding key press and the visual and/or auditory change on a computer monitor at a later time when performing the originally viewed action. This action could be a tricky manoeuvre in a video game, or a certain setting within an options menu. Further discussion around this topic support this view in the article 'Role of action observation and action in sequence learning and coding' (Boutin et al, 2010, 250) where the paper suggests "movement sequences acquired through observational practice are effector-independent and are coded in visual-spatial coordinates" which in theory the research should directly relate in such a way that the movement sequences are similar to several key presses to perform an action that is observed online, which during viewership will in theory be coded in the brain through visual-spatial memory.

Without the ability to know what keys were pressed at given moments in time as a response to visual and auditory stimuli, the parallel process Hikosaka et al propose might not begin unless the viewer was already fully experienced with the content of the video. E.g. a professional e-sports player watching another professional e-sports player (without a key overlay) would be much more successful in associating key presses due to their visual stimuli association of the action and its corresponding key press. Therefore, it would be beneficial to have the ability to display a stream of key presses to viewers to learn through observation visually. (Boutin et al, 2010).

To further support the aim and previous discussions; the results and summary from 'Distinct modes of executing movement sequences: Reacting, associating, and chunking' (Verwey and Abrahamse, 2012) show evidence of improved reaction times and the performance of participants through familiarisation of the tested sequences. Participants were also asked to repeat the sequence after the initial test, indicating that more people succeeded in recognising the sequence rather than recalling from memory. This combination of repeated exposure to a sequence and the increased ability to recognise that sequence visually provides promising grounds to develop this plugin in such a way that aids the chunking of key presses into sequences associated with goals.

These articles provide very positive reasoning for such a plugin to exist but there are certain aspects out of the scope of this project that need to be discussed. One of which is mentioned in 'Learning a keying sequence you never executed: Evidence for Independent associative and motor chunk learning' (Verwey and Wright, 2014, 29) which discusses that learning keying sequences from observation association is improved when accompanied with motor chunking (physically using and learning the key presses overtime). This indicates that over time the key overlay may become less useful to viewers who watch only one person, as the viewer will eventually become accustomed to the observation and will later in learning move to a motor chunk method of recalling the key sequences. It is not to say however that this is a negative point as recalling movements through

motor chunking Is often shown to be faster in research results (Boutin et al, 2010; Verwey and Abrahamse, 2012; Verwey and Wright, 2014) which have been learned over time through physical practice. Therefore it may be recommended to users of this plugin that actual practice of the observed key sequences will increase the rate of procedural learning.

Many of the discussed studies are geared around very specific tasks and learning methods which may not necessarily transfer to other skills or events in life. However, an interesting discussion can be found on 'Exercising Your Brain: A Review of Human Brain Plasticity and Training-Induced Learning' (Green and Bavelier, 2008) in which Video Games have shown to have beneficial effects to those who play Video Games that require fast reactions and their full attention (Green and Bavelier, 2006). It is clear that the act of playing active video games can improve several aspects of cognition. One of these aspects most notable for this proposal is presented in the results of 'Action video games modify visual selective attention' (Green and Bavelier, 2003) in which those who played action video games could read a sequence of letters (presented at an extremely fast pace) more accurately than those who did not play action video games. The artefact would likely need to compete with active elements within the video for screen time and the attention of the viewer, therefore it is reassuring to know that the target audience for Twitch.tv is mentally accustomed to seeing and acquiring more information at once. Conversely, those that would view these videos on YouTube for purposes other than video games would be able to pause/rewind the video unlike that of a live stream.

In contrast to the psychology and procedural learning of tasks, a good question to ask is 'How do people *actually* learn keyboard shortcuts?' which is effectively posed and answered in 'Using strokes as command shortcuts: cognitive benefits and toolkit support.' (Appert and Zhai, 2009). This research paper takes the user behaviours perspective to assess how shortcuts and Graphical User Interface design can aid recognition and memory recall of shortcuts. Throughout the paper, 'stroke shortcuts' (similar to gestures and shapes as inputs) are discussed as more effective than keyboard shortcuts due to their real world relatability for users as they can recall their motor movements and what the 'stroke' reminds them of in a personal way. This indicates that the design of the plugin and key overlay would benefit from making these shortcuts and key presses more familiar or 'openly-interpreted' to the viewer, so that they may make their own visual-spatial memory associations.

3.2. Market and Existing solutions

The online entertainment market has surged greatly over the last decade and it is worth expressing the potential viewer (and user!) base in relation to these statistics. According to Twitch.tv themselves their unique viewership more than doubled to 45 million people per month, 12 billion minutes watched every month across a total 6 million videos. (Twitch.tv, 2013, 5). Twitch.tv isn't the only place that has seen internet video entertainment explode. YouTube has continued to rise in popularity and boasts that people watch 'hundreds of millions of hours' per day, alongside stating they have more than 1 billion users. (YouTube, 2015).

To better emphasise the movement to online entertainment Twitch.tv state that "68% have decreased watching TV to focus their time on game entertainment" (Twitch.tv, 2013, 9). This suggests that further developments on making the most of online entertainment from places such as YouTube and Twitch.tv are going to become more and more contested as viewership rises alongside business interest. It is no surprise then that the open source community (OBS Studio) are heavily involved in developing tools to support this entertainment boom.

It has been noted that for some dedicated communities to particular software or games, viewers demand to be able to see both the subject (player/user) and keyboard with separate webcams.

(SpareOsu, 2015). Which is both monetarily and computationally expensive for the user, a plugin that could keep up with this demand whilst still being accepted by its community would be ideal.

There are a few similar solutions to the proposed problem that already exist, yet none of them seem to be strictly plugins. All of the solutions proposed are standalone programs.

Figure 3: Screenshot of StreamKB developed by Mikecustomcode.

StreamKB as seen above (Mikecustomcode, 2013) is an open source program written in java that creates a very small keyboard that displays system wide key presses on the fly. The design of the keyboard itself is very clear, yet saves space and thus does not clutter the screen. The user is expected to use OBS to capture it on screen or make the keyboard 'always on top' of other windows for capture. Whilst this program is well written in java and handles fast key sequences with ease, it hooks into Windows itself to pull back every key press regardless. This re-raises the concern of a user typing in their password on a livestream or recorded video and any viewer could then steal that password.

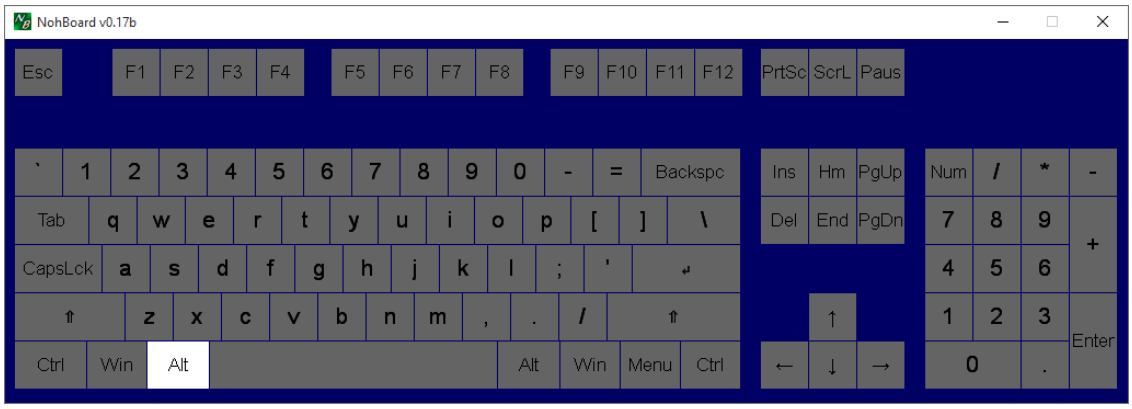

Figure 4: Screenshot of NohBoard developed by TheNohT.

NohBoard, above, developed by (TheNohT, 2015) is another open source standalone program, but is written in C++. This keyboard program has a more typical keyboard layout and includes extra function keys and the number pad, however the design is much larger and could cause some obstructions during viewership. However, NohBoard does have the ability to hide the title bar and due to the flat unique blue background OBS and other streaming/recording software can key out that blue to make it invisible (in the same way a green screen works) during recording.

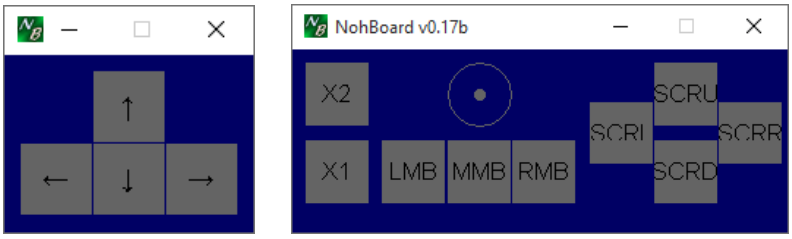

Figure 5: Screenshots of two more layouts provided by NohBoard.

Uniquely, NohBoard has several keyboard 'layouts' as seen in Figure 6, one of which is for mouse capture. The capture is analogous to an analogue stick on a controller, this will show direction and travel but not acceleration or position of the mouse cursor. Whilst these particular layouts may not be of interest, the idea of creating custom layouts for users so that passwords would never be entered due to the restriction of keys is a potential solution to the ethical problems presented.

Figure 6: Screenshot of Gop Gop Goop Goop developed by Sfkjhhdslu

Gop Gop Goop Goop is developed by (Sfkjhhdslu, 2013) and again is a standalone program. This program is written in C# and provides a lot more functionality over StreamKB. Features include disabling of individual keys, several mouse buttons and clicks. However, it is much larger than StreamKB and has no configuration on the window size or layout.

3.3. Summary

To summarise the review of literature and existing software;

- Procedural learning does not have to be sequential
  - Parallel process of learning (Visual-spatial)
- Familiarity and User behaviour are important to recalling keyboard shortcuts
- Existing solutions are solely standalone programs
- Existing solutions do not support the discussed methods of learning effectively.
  - Via Observation and Visual-Spatial memory.

As a final point, all of the keyboard programs discussed are not overlays and all of them only have US ANSI QWERTY keyboard layouts, which is okay for Video Games due to the keys and their placement typically used are standard to all keyboards. This would negatively affect software tutorials on recorded videos as key placement for shortcuts and hotkeys will be in different places as well as on different keys. This change of keyboard layout also interrupts the visual-spatial memory and observation process of learning due to the different locations of keys, which may need to be tackled in the plugins design.

4. Project Plan

4.1. Development Life Cycle

In order to properly plan out the development of this artefact, the System Development Life Cycle (SDLC) will be discussed in relation to this project. The SDLC outlines several parts;

1. Requirement Analysis

Requirement analysis is a crucial part of ensuring a system or software is designed and implemented correctly, as without knowing what the client/customer/user requires makes the following steps impossible to do correctly. Within this project, the artefact will be based upon requirements outlined by the aim as well as those that arise from users, testing and further analysis or research.

2. Design

The Design of software or systems is critical to its usability, functionality and simplicity. Good designs are often simpler ones with all the required functionality (or more). The design stage will certainly be incorporated into the project and possibly re-designed after iterations (See 5. Evolution).

3. Implementation

The implementation of software or a system is critical to meeting the requirements outlined. The implementation will be carried out several times after testing and further requirements and/or design changes. Thus, ensuring that the implementation is the correct one for its users.

4. Testing

Testing is a necessary part of the life cycle, in that it is used to truly know whether the design and implementation have any merit. The artefact will be tested after iterations of development to ensure the design and implementation are on course and to provide valuable feedback for the report to follow.

5. Evolution

Whilst technically out of scope, the evolution of this artefact will be somewhat taken forward from the iterations developed and tested, but not necessarily evolved due to potential feature creep and time constraints. It is more beneficial to stick to the aim and domain of this project. However, there may will be discussion for further evolutions beyond the scope of this proposal and project that the Open Source community take forward.

4.2. Development Methodology

As the discussion of the SDLC above points out (and throughout the objectives outlined in section 2), it is clear that users need to be involved in the development process. This somewhat rules out a typically Waterfall style of development, due to its design and test at the end method. An Incremental or Evolutionary methodology would bring more emphasis and feedback to the development of this artefact in such a way that earlier iterations and a user focused design will lead to a plugin that is geared more towards an open community. (Cotton, 1996). Thus the following sections outline a suitable plan that incorporates time for a ‘release’ and further ‘iteration’.

4.2. Gantt Chart

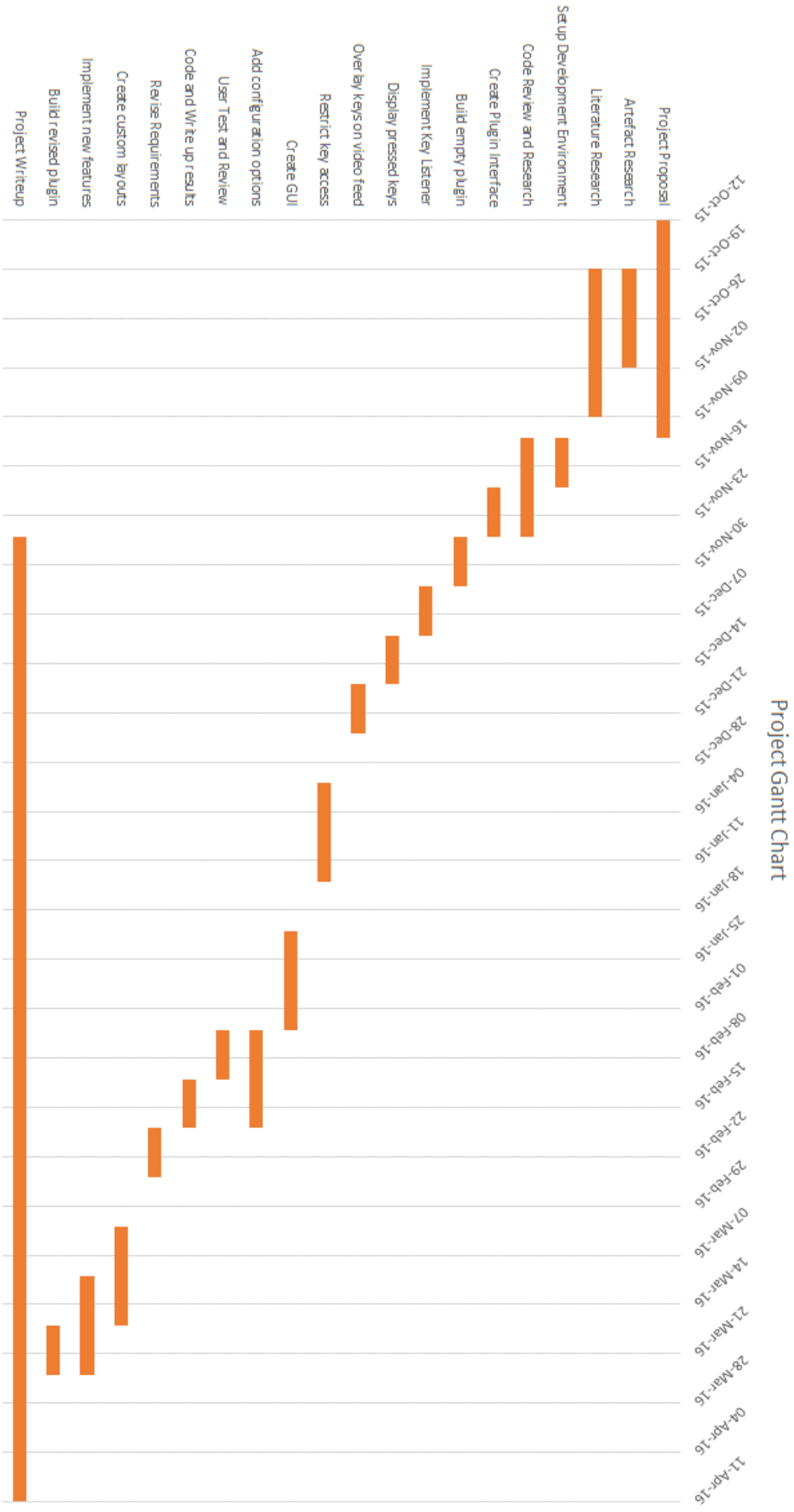

Figure 7: Gantt Chart detailing the high level tasks required to complete the project.

4.3. Gantt Chart Milestones

The Gantt chart in section 4.2. (Figure 7) is plotted in terms of weeks. A 'week' will realistically be around 24 hours contributed time towards the project and not a typical work week. With that in mind, discussing the milestones and tasks associated from the chart will outline the realistic and achievable plan.

There is also some 'downtime' which are both buffers and time off, such as Christmas and further important time periods such as exam weeks and later hand-ins. These were planned around to mitigate risk of

Milestone 1 *- Project Proposal, Artefact Research, Literature Research*

Clearly, this milestone has already started and is well underway with most sections completed. However it was added into the chart to help ensure the bigger picture and full plan is outlined. This milestone needs to be completed not only due to deadlines but is required in order to fully understand the first iterations requirements, methodologies and aim.

Milestone 2 - *Setup Development Environment, Code Review and Research, Create Plugin Interface, Build empty plugin*

This milestones represents the first stage in development, which is required to be completed before features can be implemented. The close nature of each task makes this milestone very achievable in the time frame scheduled in Figure 7 and will aid in the understanding of the underlying elements of the design before implementation is considered.

Milestone 3 *- Implement Key listener, Display pressed keys, Overlay Keys on Video feed*

Once development has been well established, this milestone will begin what will be the first 'iteration', with an aim to create a very basic display of key presses on video, with no consideration to visual design. This will ensure that the outcome of this milestone is to get the functional aspect of the plugin correct before configuring the design and advanced features to comply with further requirements and learning methods as discussed earlier.

Milestone 4 - *Restrict key access, Create GUI, Add configuration options*

After the first milestone has been completed, there is some buffer time to allow for development to take longer than expected in case of unforeseen issues. This buffer may contribute time towards holidays, milestone 3 or 4 depending on the need at the time. This milestone is aimed at fleshing out the plugin to fit the aim and project title, with a large focus on solving the ethical issues outlined.

Milestone 5 *- User Test and Review, Code and Write up results, Revise Requirements*

On completion of Milestones 3 and 4, the plugin should be ready for a User study/test to elicit its performance and revise requirements for a second iteration as outlined in the software development methodology. This review of the plugin will ensure that the delivery, design and implementation are contributing towards the aim of the project.

Milestone 6 - *Create custom layouts, Implement new features, Build revised plugin*

Following the completion of milestone 5, further advanced features and feedback from the user study / revised requirements can be worked on. Due to the unpredictable nature of the revised requirements and feature list, a couple of buffer weeks have been placed at the end of the

development time frame. This helps to ensure reliable time management and planning for variable circumstances.

Milestone 7 - Project Write-up

Finally, the project report needs to be written alongside most of the development processes. The buffer time at the end of the project is planned for use for completing development as well as spending more focused time completing the documentation. This should ensure a close link between the developmental process, issues, feedback and research with the write up.

4.4. Risk Matrix

Risk	Likelihood	Impact	Mitigation
Personally being ill or unable to work.	Low	Due to the nature of this project being predominately desk based, the impact will be minimal.	Ensure work is not done in bulk, but progressively each week.
Delays and issues to setting up Plugin Development.	High	Until this hurdle is overcome, it could have a medium-high impact on the start of the project.	Abundance of open source assistance available and University staff to assist if needed.
Supervisor unavailable	Moderate	As the project spans for several months, it is moderately likely that the supervisor may be unable to attend a meeting or respond appropriately due to illness.	Ensuring that work moves at a consistent steady pace so that any one meeting does not become more critical than another.
OBS Studio Development Issues	High	Slowed development and harder implementations of features and cross-platform compatibility. Moderate impact.	Target only Windows initially to ensure at least one version works. Join the Open community as documentation is lacking.
CMake Development issues	Moderate	Slower initial development until it has been setup. Minimal impact.	Assistance with Supervisor and online Documentation to speed up the process.
Problematic implementation	High	It is very likely that implementing a particular function/feature will be almost impossible due to how new OBS Studio is, and that it is not feature complete.	This will have to be mitigated by developing workarounds or simply putting features on hold until development progresses.
Home computer breaks.	Low	Low impact to work however the schedule and plan may need to adapt.	GitHub repository and OneDrive will keep independent copies to ensure there is no data loss to work, including documentation.
Updates to OBS Studio cause issues to plugin	Moderate	Moderate impact, unforeseen updates may create new roadblocks for development.	If this becomes too big of an issue, development may have to be kept on one version until OBS matures.

Unable to implement full plugin within OBS Studio	Low	High impact, the project will have to target another version or stand alone.	If this occurs, then development will need to target an old version of OBS or write a standalone program.
Loss of work.	Low	High impact.	All associated work (including the artefact) will be stored on a private repo on GitHub and OneDrive.
Missed objective deadlines.	Low	Moderate Impact	Development for certain objectives may take longer to implement due to bad estimations or unforeseen issues, avoiding these where possible.
Final product does not fully achieve aim.	Low	High Impact to success of plugin, but will still yield useful results.	To prevent the final product from not fulfilling the aim confidently, testing and research will be done frequently as iterations are created.
Project Scope is too great	Moderate	High Impact as the measurable objectives could be impossible to meet due to unforeseen issues or poor planning.	To prevent this, a large amount of planning and research has already been done before the completion of the proposal.

5. References

Appert, C. and Zhai, S. (2009) Using strokes as command shortcuts: cognitive benefits and toolkit support. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* ACM, 2289-2298.

Boutin, A., Fries, U., Panzer, S., Shea, C.H. and Blandin, Y. (2010) Role of action observation and action in sequence learning and coding. *Acta Psychologica,* 135(2) 240-251.

Cotton, T. (1996) Evolutionary fusion: A customer-oriented incremental life cycle for fusion. *Hewlett Packard Journal,* 47 25-26.

Green, C.S. and Bavelier, D. (2008) Exercising your brain: A review of human brain plasticity and training-induced learning. *Psychology and Aging,* 23(4) 692-701.

Green, C.S. and Bavelier, D. (2006) Enumeration versus multiple object tracking: The case of action video game players. *Cognition,* 101(1) 217-245.

Green, C.S. and Bavelier, D. (2003) Action video game modifies visual selective attention. *Nature,* 423(6939) 534-537.

Hikosaka, O., Nakahara, H., Rand, M.K., Sakai, K., Lu, X., Nakamura, K., Miyachi, S. and Doya, K. (1999) Parallel neural networks for learning sequential procedures. *Trends in Neurosciences,* 22(10) 464-471.

Verwey, W.B. and Abrahamse, E.L. (2012) Distinct modes of executing movement sequences: Reacting, associating, and chunking. *Acta Psychologica,* 140(3) 274-282.

Verwey, W.B. and Wright, D.L. (2014) Learning a keying sequence you never executed: Evidence for independent associative and motor chunk learning. *Acta Psychologica,* 151 24-31.

GitHub Inc. (2015) Github Where Software Is Built. [online]. Available from https://github.com/ [Accessed 06 Nov 2015].

Microsoft (2015) OneDrive – Cloud storage from Microsoft. [online]. Available from https://onedrive.live.com/ [Accessed 06 Nov 2015].

Twitch.tv (2013) Twitch 2013 Retrospective. [online]. Available from http://www.twitch.tv/year/2013 [Accessed 07 Nov 2015].

YouTube (2015) Statistics – YouTube. [online]. Available from https://www.youtube.com/yt/press/en-GB/statistics.html [Accessed 07 Nov 2015].

SpareOsu (2015) SpareOsu Playing Osu! [online video]. Available from http://www.twitch.tv/spareosu/v/23311853 [Accessed 07 Nov 2015].

Mikecustomcode (2013) StreamKB [software] version 1.3. Available from http://sourceforge.net/projects/streamkb/ [Accessed 07 Nov 2015].

Sfkjhhdslu (2013) Gop Gop Goop Goop [software] version 0.12. Available from http://sourceforge.net/projects/gopgopgoopgoop/ [Accessed 07 Nov 2015].

TheNohT (2015) NohBoard [software] version 0.17b. Available from https://github.com/ThoNohT/NohBoard [Accessed 08 Nov 2015].