User Centred Design

Assignment 1 – Usability Guidelines Report

Educational Programming Software

An Extended Usability Study on Educational Programming Software Aimed At School Aged Individuals

Table of Contents

1. Introduction pg.4

2. Current State Of Educational Programming Software Pg.4-5

3. Current Design Trends Pg.5-6

4. Current Design Guidelines Pg.6-9

4.1. User Needs And Expectations pg.6-7

4.2. User Interaction pg.7-9

5. Interface Design Pg.9-11

5.1. Reactive Screen Size pg.9

5.2. Minimalistic Design pg.10

5.3. Navigation pg.10

5.4. Consistency pg.10

5.4. Error Handling pg.10-11

6.Justifications pg.11-16

6.1. Durable Interface All Screens pg.11-13

6.2. Reactive Screen Size pg.12-13

5.2. Minimalistic Design pg.13

5.3. Navigation pg.13-14

5.4. Consistency pg.15

5.4. Error Handling pg.15

7. Conclusion pg.16

8.Bibilogrpahy pg.17-18

1. Introduction

As we progress even further into the digital age the constant need and demand of higher levels of digital literacy only grows and grows as every year passes. Following this shift we have seen a new digital curriculum taking effect throughout schools that focus's more on programming now rather than Microsoft Word and Microsoft Excel according to the Australian Curriculum (2019).

With this new focus on programming a brand new range of educational software has to be developed to be able to be used by individuals as young as '4 years' (The Department of Education Tasmania, 2019) and as old as "70 years" (Victoria University, 2016) in some cases within the Australian public and private school system. As a result brand new software is required because currently the software on the marketplace is designed towards only a single demographic, which are either young children using block coding or adults programming in specific languages like C++.

Due to this wide-ranging age gap a specific set of design guidelines have to be implemented in order to ensure that the software is usable by individuals of all ages within the school system. However at the same time the software still needs to contain enough functionality to allow the students to program as fluently as you would in visual studio code or Scratch to ensure we don't create a negative user experience through the loss of functionality.

Nevertheless it also needs to be noted that the software has to be adaptable towards the users current learning needs and wants. This is primarily because individuals learn at different rates and with different learning styles (Felder, 2002, p.4). This poses a significant issue for designers trying to implement this into the design without compromising other aspects of the design to include this functionality for the user.

2. Current State Of Educational Programming Software

Currently on the marketplace there exists a multitude of educational programming software such as Scratch, SoloLearn and PluralSight which all teach individuals how to code in different ways. However a large majority of them focus on a single demographic. These demographics specifically being:

1. Children – Learning to program/Code with block coding as seen in Scratch
2. Young adults – Learning specific languages such as C++ as seen in SoloLearn or Plural sight

Due to these two distinctive demographics requiring very different design features and functionality's very few applications exist which include both block coding and the ability to program in multiple languages effectively. However one piece of software that integrates the two effectively is an application called Choregraphe.

Choregraphe is the graphical user interface designed by Aldebaran robotics to allow users to control their Nao and pepper robots. The software use's a mixture of block type coding paired with the option with to code in either python or C++ to control the robots. Choregraphe is an application currently in use by both primary schools and high schools within Australia to teach children programming and robotics according to the Australian Curriculum (2019)

However while there might be a limited source of ideal software to review, when broken down the majority of software on the marketplace aimed at a single demographic all follow the same hidden list of rules/professional practice as to how best layout their product to ensure the user gets the most of out the learning experience.

3. Current Design Trends

Whether the software is designed towards children or adults or both children and adults, both applications still follow the same design trends when laying out their specific software.

However it needs to be noted in particular that these trends will most likely change as this type of software is designed around users and when the user's needs change so will the software. As a result users currently require a more personalised learning experience tailored towards their needs and wants. Nevertheless a single constant that has stayed the same is that they have always had a minimalistic and familiar design pattern for finding key information. The most common and relevant design trends are shown below:

- Use of Fitt's Law - Almost every large scale educational software and most software today implements Fitt's Law into it's design to ensure key information such as the registration, home button and help section is easy to find and use (Tang et al. 2017, p. 33-40). They do this to ensure the software has a high level familiarity when a new user starts to use the software. Which in turn results in the user spending less time looking for certain functionality's. Resulting in not only a better user experience but also more effective and meaningful human computer interaction. Ultimately resulting in the user learning more in a shorter period of time.
- Personalised account feed - The majority of educational software in the modern age has a distinctive and adaptive account feed where the user is able to view their current progress. In which they are able to see how they are going and where they might need to improve based upon the answers and tasks they have completed with the software. This helps to ensure that the user is focusing primarily on the area's
- Colour – The majority of educational software on the market place followed a specific colour scheme using colour theory generally focused around lighter pastel colours and only ever used neutral colours when displaying images of varied colours. Most educational software uses this logic paired with applied behavioural analysis to ensure a fluent and aesthetically appealing design to ensure the software is easy on the user's eyes which in turn results in the

software being less strenuous on the user's eyes and decrease's the likelihood of the user become angry of frustrated at the software.

- Embedded teaching videos. Most of the educational software had embedded teaching videos in there software which appears after failing a specific question more than three times that pop's up. This embedded teaching video would then explain the current aspect being taught whether it be variables or a depth first searching algorithm. Generally it is a professional video in which the user is able to interact with it to ask questions to the pre-recorded video.

4. Current Design Guidelines

Based upon the vast differences in functionality and requirements needed within both a child's programming software and a young adults programming software a specific set of guidelines have to be adhered too to ensure no key elements are left out during the designing phase when making software aimed at both demographics. For that reason we will be hypothetically classifying them into the:

- Usability framework (Nielsen, 2003)
  - Learnability
  - Efficiency
  - Memorability
  - Errors
  - Satisfaction
- Usability Constraints (Mayhew, 2006)
  - Cognitive, perceptual and motor capabilities
  - Users' physical and social environment
  - Software and/or hardware constraints, and system platform

4.1. User Needs and Experiences

As mentioned above at the heart of designing educational programming software the user is the key focus and has to be constantly evaluated before the design as the users needs and wants will constantly change. Multiple studies get conducted/re-evaluated every year to ensure how best to educate individuals. One such study conducted by Julie Dirksen called Design for how people learn (2015) emphasises how individuals learn best from a combination of textbook theory work combined with another teaching assistance such as an individual with extensive knowledge on the matter guiding them through the work.

However it also needs to be noted that not only does Dirksen also suggests another external learning factor but also that you have to design the learning software in order to predict:

- The individuals current knowledge level
- The individuals current skills in the particular area
- When you need to motivate the individual to keep studying
- When best to integrate a individuals daily habits into their own study routine
- The current environment the individual is being taught in

These design factors all have to be taken into careful consideration when implementing the design of any educational software as if one is overlooked it can not only possibly cause the individual to lose interest and have a negative user experience but it can more seriously set off a spiral effect in which they eventually lose all motivation to study.

However since we are dealing with individuals of an extremely young age who might have an extremely low level of digital literacy we have to ensure we place emphasis on the familiarity of the software but also the languages the software supports as well as how we convey knowledge towards the users. This is particularly important within Australia as it becomes even more multicultural within the school system as there is now a high likelihood that English could not be an individual's first language.

Furthermore we need to take into consideration how the software will adapt to individuals with either a physical or cognitive disability. As these need to be handled with the upmost care and importance as not to discriminate against anyone as the aim of the software is help individuals learn programming of all capability's.

4.2. User Interaction

A users interaction within all educational programming software can be primarily broken down into the following screens:

- Home Screen
- Graphical User Programming Screen
- Progress Screen
- Help Screen

As a result these four screens requires the user to drastically do different things on each screen. For that reason these four screens have to be considered in their own category as to ensure no user interaction is missed or overlooked during the design and usability phase.

4.2.1. User Interaction – Home Screen

When designing the home screen of the software we need to remember that we can possibly be dealing with individuals of an extremely young age with low levels of literacy. For that exact reason we have to ensure that the login design is fluent and clear to point as many individuals using the software might have a low level of digital literacy.

For those exact reasons the user interaction design guidelines on the home screen will encompass the following elements:

- The integration of a login system where the user will be able to recover their username and password easily if they would forget. Ideally the software will automatically login the user if it is not there first login attempt. This is to ensure individuals are able to login straight away and not have to deal with logging in if not needed.
- An error message will have to be provided towards the user in layman's terms in the specific language they have selected with a solution added onto it as to how successfully login and how to recover their account if they have forgotten either the username and password or both.

- The ability to change the language the software is currently is in. This is to ensure that individuals who do not have English as their first language are still able to use the program regardless of the language they speak most fluently.

4.2.2. User Interaction – Programming Screen

When designing the graphical user interface that will allow the user to program within the software we have to consider a multitude of factors on how the user will interact. The most common ways the user will interact with the software are:

- The integration of a block type like programming system and a normal programming language together in one without losing functionality or readability. When designing this we need heavily think about how the different demographics will use this screen. The younger demographic will only be using the block type programming system while the older demographic will be using the more complex functions. For this we need to consider the effective use of white space and drop down bars. This is to ensure we don't overwhelm the user's specifically the younger demographic.
- The box libraries for the block programming need to be fully explained layman's terms but still have the technical knowledge not taken out of the software. This is to ensure that the user gains the technical knowledge while still understanding it in plain English.
- The user's needs and wants of specific functions need to be flexible as so the user can takeaway or add functions that they want or don't want. This is to ensure that the user can use the software fluently and engage with the software to its full extent.
- The error messages that pop when the user incorrectly does a task has to be clear and straightforward to the point explained in both layman's terms and technical knowledge. However it also has to have a direct point and direction towards the user as to how to fix the issue.

4.2.3. User Interaction – Progress Screen

When thinking about how the user will interact with the progress screen we have to realistically realize that they are only going to be viewing the screen to see how they are progressing. For that reason we need to ensure that this screen only provides all relevant information they are looking for. This information would be however not limited to:

- All relevant information used to describe how the user is currently progressing has to be shown and presented in a structured format. This is to ensure that the user is able to intelligently interact with the progress screen and see what they need to improve on.
- All relevant information on the screen is described in either plain English or layman's terms with the exclusion of technical terminology. This is so that the user knows where exactly and what specific tasks they need to revisit to better learn the theory being taught.
- Apply Applied behaviour analysis when considering the colours to apply to the graphics used to demonstrate the user's progress and current success as to ensure that we don't abruptly cause the user's mood to negatively change based upon the viewing of the graphics

4.2.4. User Interaction – Help Screen

When designing around how a user interacts with a help screen we need to go back to our baseline guidelines from the usability framework (Nielson, 2003). Through following this specific usability framework we are able to almost perfectly plan and predict almost every scenario the user will use this screen for. Some of the design considerations needed to noted are:

- Applied Behavioural analysis has to be included with the design to ensure when the user receives an answer that they are not looking for that it helps reassure them that the answer given is correct and the right path to take
- The language used to convey the answer and the help aimed towards the user has to contain no jargon, which might confuse them. As to ensure that the help is well received and understood.
- The way the user receives the information has to be in a structured and logical format. This ensures that the user is able to actively take in the information presented and process it to make a decision as to how best to use it.

5. Interface Design – Durable Interface For All Screens

When designing the interface and thinking how the user will interact and change the interface we need to ensure that it doesn't lose any functionality while the user does this. However at the same time we need to ensure that the user doesn't change the interface so much that it removes all functionality and use as an educational piece of software.

5.1. Interface Design – Reactive Screen Size

Within the current evolutionally trend of technology the size's of screen change constantly and rapidly. This is particular evident when some user's might use this software on there phone while others on their desktop pc. For that reason we need to ensure that design of the software change's based upon the user's screen size. By ensuring the software has a reactive interface design it ensures that all elements on the screen are perfectly placed and maintain their structure.

Which not only helps organize all information in a logical order on the screen, however it also contributes to a more fluent flow of how the information is organized contributing to a lowered reading time. Which in turn ultimately results in a greater user satisfaction with the software due to how structured and fluent the information provided is.

5.2. Interface Design – Minimalist Design

Due the software including a multitude of information and functionality's we need to ensure that this excess of information and functionality does not negatively impact the design. To ensure that we need to make a clear point to not confuse new users especially those with a lower literacy level. We do this by ensuring we have a minimalist design paired with various drop down menus. This ensures that the program does not overwhelm new users, however while at the same time ensuring that the pre-existing users have all the tools needed. This is very important as various research papers and journals have continually shown that the majority of young children become overwhelmed easy. This is the same with some adults depending on the various factors such as their early childhood experiences and current level of independence.

5.3. Interface Design – Navigation

Currently within educational software they can either have very simple or very complex navigational design systems. However very many of small scale educational software fail because they don't make use of Fitt's law. Which in turn results in a lower level of familiarity on how to best traverse the software.

Through Implementing drop down menu's to allow the user to navigate through the software not only does it reduce the amount of clicks required and searching to find a specific page but also helps ensure that the interface maintains its minimalist design as to not overwhelm the user's. Furthermore it helps actively engage the user, as the software will feel more responsive to their current needs.

5.4. Interface Design – Consistency

Consistency is key towards an effective and usable interface design. This is primarily because if the interface does not have consistency across all pages then the software will lose familiarity because the user will look in one place for something and it will be somewhere else. Ultimately resulting a lower level of user satisfaction and less meaningful human computer interactions.

For that exact reason the interface that is designed has to highlight specific area's of importance such as where the main navigation buttons are always shown, how much white space is always shown as well as where the additional functionality's are always shown

5.5. Interface Design – Error Handling

Error Handling and predicting a user's actions is key to making an effective, sustainable and usable design. This is because if you don't predict a user's key software breaking actions then they are forced in almost every case to hard restart the software. For that reason we need to anticipate and predict how a user might break the software.

We anticipate this and prepare against this through ensuring that we structure the information and key buttons in our software with essential error handling such as asking if the user wants to quit the program and warning them when progress might be lost. It also needs to be noted that whenever communicating key information to the user that we only ever explain information in layman's terms as to not confuse the user.

Through including this design factor it ensures that our software has a lower likelihood of shutting down unexpectedly for the user, which in turn results in more meaningful and efficient human computer interaction.

6. Justification

Through following the current trends and guidelines outlined above you would be able to make an educational software platform that not only actively engage users but also follow the current trend in today's marketplace. However it needs to adhere to that these guidelines are designed and aimed at today's marketplace and not the future or the past

Thus is has to be concluded that some of these guidelines will most likely become obsolete in the future however the majority will stay relevant as the trend and standard in design has not changed drastically in the past few decades. This can be seen very clearly through Microsoft Word's menu bar before Vista compared against Google Docs' menu bar in 2019 as shown below in figures 1 and 2.

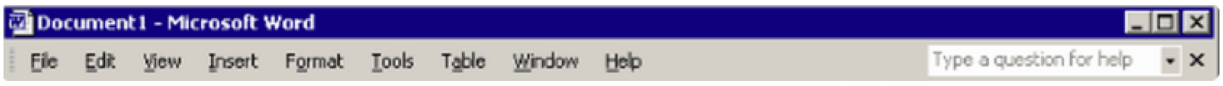


Figure 1: Microsoft Word Menu Bar (2005)

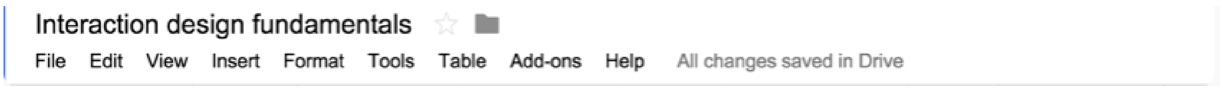


Figure 2: Google Doc's Menu Bar (2019)

Thus it can be noted that these design trends mentioned above will most likely not change because by changing them you are at a high risk losing user familiarity because they expect specific design elements follow the standard and when they don't they get confused. Which in turn ultimately causes them question and go looking as to where these elements are, ultimately resulting in inefficient user computer interaction.

6.1. Justification – Durable Interface – All Screens

It is essential that when creating the basic interface that will run over all pages of the software is that we keep a consistent look throughout. This is for a number of reasons that range from:

- Keeping user familiarity as high as possible
- Keeping user experience at as a positive when navigating
- Ensuring that meaningful human computer interaction occurs while navigating the software

One such software that does this is effectively is Scratch. As shown below in figures 3, 4 and 5 you can clearly see no matter what page of Scratch you are on the main interface does not change.

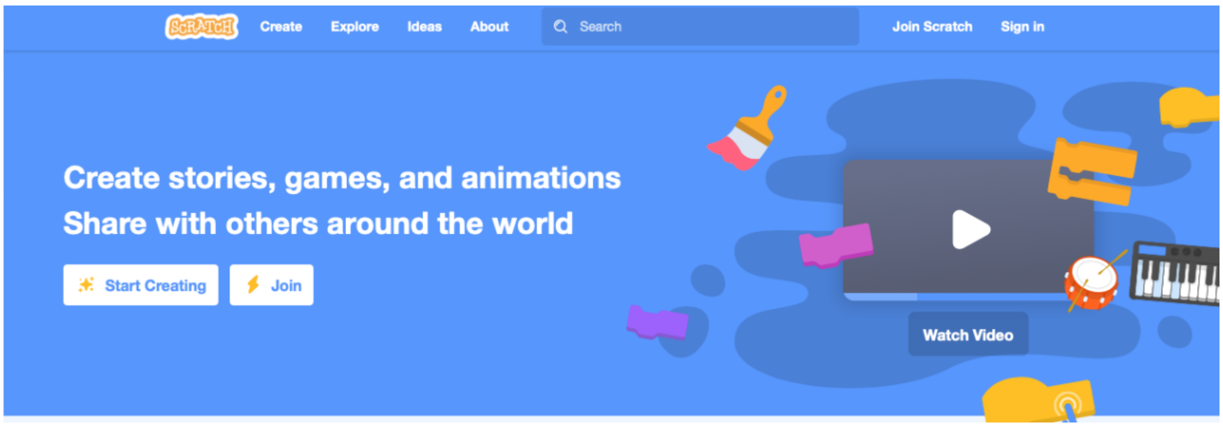
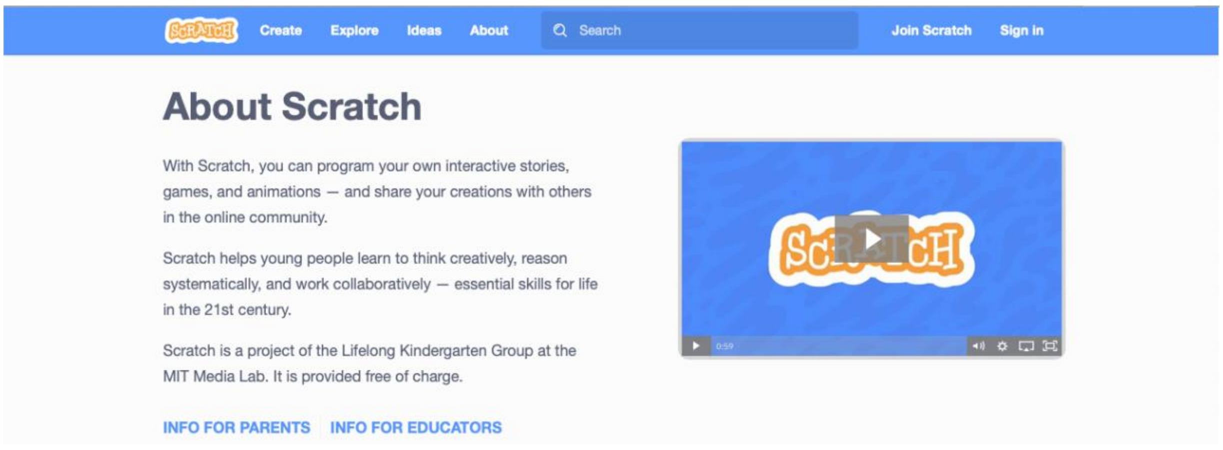


Figure 3: Scratch Home Screen View (2019)



About Scratch

With Scratch, you can program your own interactive stories, games, and animations — and share your creations with others in the online community.

Scratch helps young people learn to think creatively, reason systematically, and work collaboratively — essential skills for life in the 21st century.

Scratch is a project of the Lifelong Kindergarten Group at the MIT Media Lab. It is provided free of charge.

INFO FOR PARENTS | INFO FOR EDUCATORS

Figure 4: Scratch About Page View (2019)

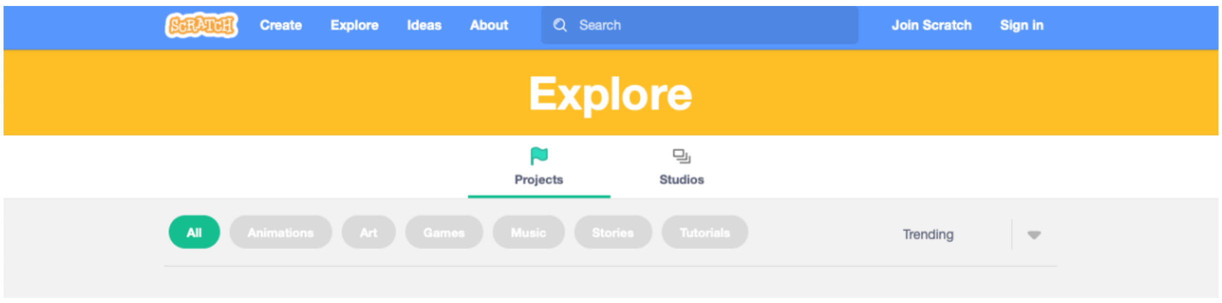


Figure 5: Scratch Explore Page View (2019)

Through Scratch having there layout like so across all page's it ensures that the user always knows where to look if they need to go either back to the home page or to another specific webpage.

For this exact reason this is why a durable interface design is a required design guideline as the basic layout of for the software. This is because no matter where the user might go on the software this ensures that they know where to go to get back to the homepage.

6.2. Justification – Reactive Screen Size

Having a reactive screen size within any software is essential. Have a reactive screen size not only ensures that users on multiple platform screen size's are able to view the modified structured data as effectively as they would on a desktop. However it also means that they can still navigate and use the software as they normally would. Ultimately ensuring that they have a high familiarity level with the software and are able to still have meaningful human computer interaction due to the structured data being modified towards there devices current needs.

When looking at large scale company's and how they create a reactive screen size it becomes evident that almost any software requires this due to the increased functionality that it brings with it. One such company that effectively uses this method is Consumer Affair's Victoria. Were able to see this when we view the website in both an iPhone 5/SE view and desktop view as seen below in figures 6 and 7.

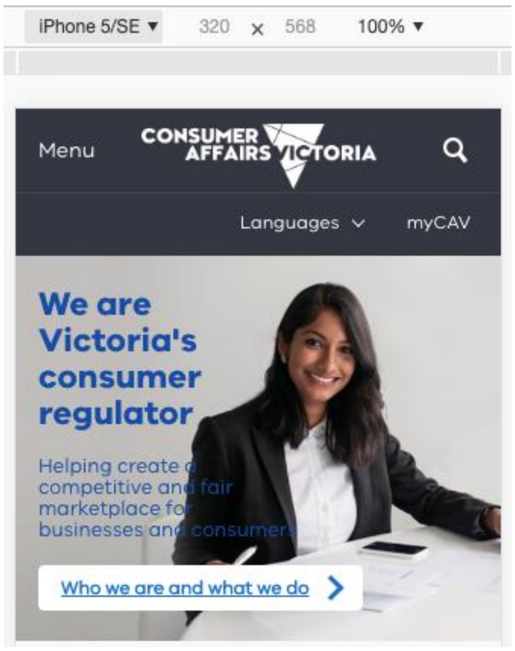

iPhone 5/SE ▾
320 × 568
100% ▾
Menu
CONSUMER AFFAIRS VICTORIA
Languages
myCAV
We are Victoria's consumer regulator
Helping create a competitive and fair marketplace for businesses and consumers
Who we are and what we do

Figure 6: Consumer Affairs Victoria iPhone 5/SE View (2019)

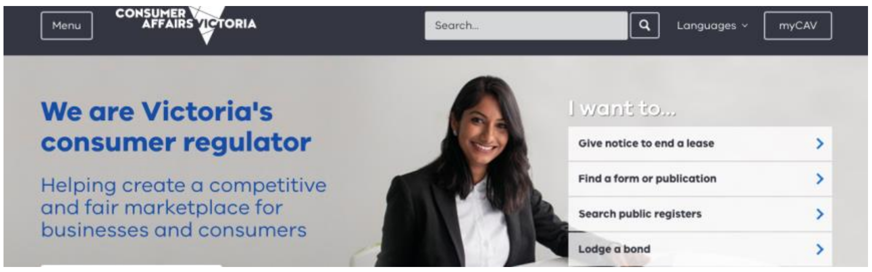

Menu
CONSUMER AFFAIRS VICTORIA
Search...
Languages
myCAV
We are Victoria's consumer regulator
Helping create a competitive and fair marketplace for businesses and consumers
I want to...
Give notice to end a lease
Find a form or publication
Search public registers
Lodge a bond

Figure 7: Consumer Affairs Victoria Desktop View (2019)

As we can see how Consumer Affairs Victoria change's their design layout based upon how the current screen it still ensures that all information is is still on the page and accessible by the user. This is why having a reactive screen size is a design requirement because it ensures that the software still maintains it familiarity and accessibility on various screen sizes.

6.3. Justification – Minimalist Design

At the heart of all educational programming software a minimalist design is currently the industry standard. This is for a number of reasons that range from ensuring the user does not get overwhelmed from the design to ensuring the workspace is large enough for the user to program in. At the core of minimalist design's drop down menus effectively ensure that the software still maintains not only its functionality but also it's navigation and consistency.

This is primarily because the user will know constantly where to look to add more functionality into the software. One piece of programming software that makes the effective use of a minimalist design is Choregraphe. Choregraphe's graphical user interface ensures that not the user has enough space to program in but also the option to add more functionality through the view drop down menu. This is displayed below in figure 8.

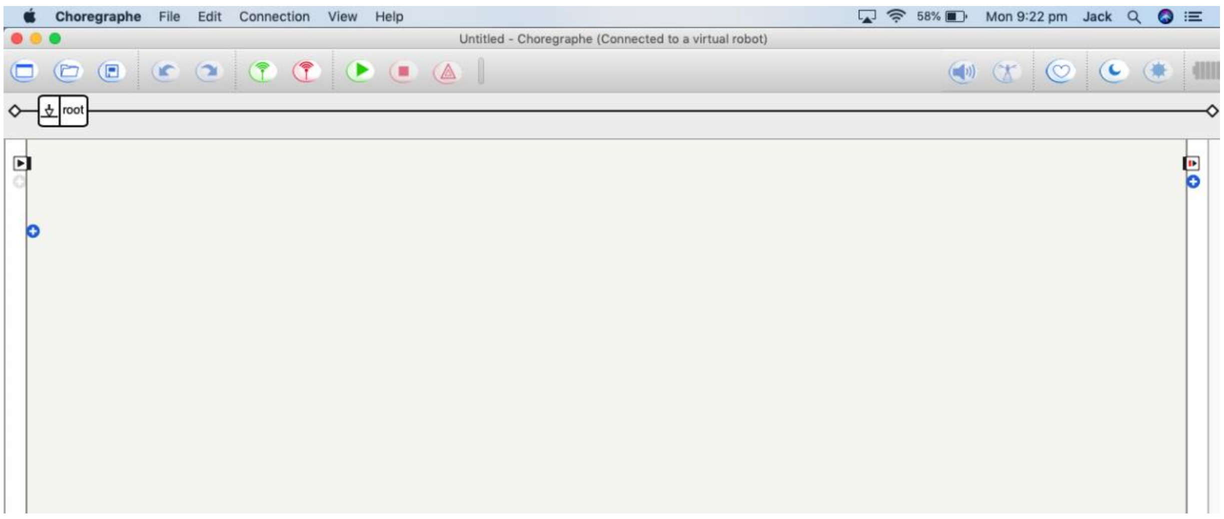


Figure 8: Choregraphe's Graphical User Interface Version 2.1.4.3

6.4. Justification – Navigation

At the heart of every piece of software is its delicate network of links that a user use's to navigate the software. However if even one of these links fail, it drastically decrease's not only the user's satisfaction of the software but the usability. Therefore it is essential that we not only place a heavy emphasis on how we design the navigation within our software but also how we implement it noting that we have to be consistent over all screens of the software.

This is primarily because as the user navigates through the software not only does it stimulate the user and create a form of meaningful human computer interaction but also increases the likelihood that the user will want to further interact with the software.

Sample navigational layouts and usability examples around various software's and websites can be seen above the current figures directly showing that not only a multitude of ways to get the user to navigate around the piece of software but also how differently user's can navigate.

6.5. Justification – Consistency

Consistency is essential to creating a high level of familiarity within a piece of software. However consistency is also the driving wedge which increase's user satisfaction and navigation within software. This is because if the user is able to find the same essential functions on various screens of the software not only does it reduce the likelihood of the user becoming angry but it also increases the user satisfaction of the software. Whereas on the other hand it needs to be noted that if consistency is not present over every page it increases the likelihood of not only decreases the familiarity within the software but also the likelihood of meaningful human computer interaction.

Therefore that reasons consistency is non-negotiable when designing software not only for educational purpose's aimed at children but in general. Examples of consistency across various pieces of software and websites can be found above in the following figures in which you are able to see how the same piece of information is in the same place no matter what happens to the size or current condition of the piece of software or website.

6.5. Justification – Error Handling

Any piece of software is able to have the most eloquent and well thought out design however if the design does not include error handling and predicting the user's actions then that is all undone. This is primarily because studies have shown that when individuals encounter issues for the large majority it increases that individual's blood pressure. Once an individual has that elevated blood pressure it only takes a few more issue's to arise until they have an extreme dis-satisfaction with the software because they see it as it does not work.

Therefore for that reason error handling is essential to include into educational programming software as if it is not included especially young children will lose all interest in using the software. One of the simplest and common error's to catch is to prompt them to save the work before they close the program. This is best shown within Microsoft Word 2019 as seen below in figure 9.

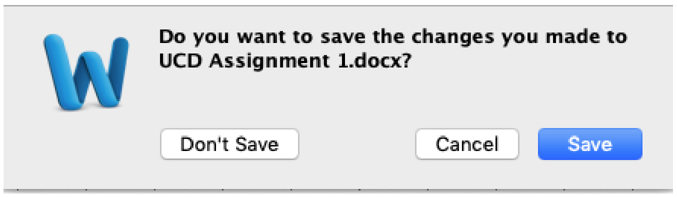

Do you want to save the changes you made to UCD Assignment 1.docx?

Figure 9: Microsoft Word's Error Catching (2019)

While we might not be able to catch every single error that might occur within the software we need to ensure that we catch the most common ones.

7. Conclusion

Ultimately, most pieces of education software are based on knowledge-acquired by the developer solely and not educational professionals. However this knowledge provided by only the developer often lacks significant relevance towards the targeted demographic whether it is children or adults. This is primarily caused by their lack of knowledge on current plight of the user. This is seen within a large number of software such as Scratch where they focus to heavily on the programming side and not enough on the user.

This ultimately cause's the user not only to lose specific functionality out of the program such as progress and current update reports but less overall satisfaction from the user as there needs are truly met nor expressed. For that reason the integration of user data needs to be included within the design phase to ensure that the current needs of the user are always met.

The inclusion of the user is especially important as there needs and wants are currently changing and evolving by the minute. However specific research has shown that the want of the user often fluctuates when new software appears or comes out.

Therefore for by designing and have a user centred design within a educational piece of programming software aimed at school age children this will ensure that not only is the software directed towards there needs but also their wants in regards to their current learning. Which in turn ultimately results in meaningful human computer interaction and but also meaningful learning for the user.

8. Bibliography

Australiancurriculum.edu.au. (2019). What do a humanoid robot and the recently awakened Narungga language have in common?. [online] Available at: https://www.australiancurriculum.edu.au/resources/aboriginal-and-torres-strait-islander-histories-and-cultures/illustrations-of-practice/what-do-a-humanoid-robot-and-the-recently-awakened-narungga-language-have-in-common/ [Accessed 18 Mar. 2019].

Australiancurriculum.edu.au. (n.d.). Digital Technologies in focus. [online] Available at: https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/?year=12986&strand=Digital+Technologies+Knowledge+and+Understanding&strand=Digital+Technologies+Processes+and+Production+Skills&capability=ignore&capability=Literacy&capability=Numeracy&capability=Information+and+Communication+Technology+%28ICT%29+Capability&capability=Critical+and+Creative+Thinking&capability=Personal+and+Social+Capability&capability=Ethical+Understanding&capability=Intercultural+Understanding&priority=ignore&priority=Aboriginal+and+Torres+Strait+Islander+Histories+and+Cultures&priority=Asia+and+Australia’s+Engagement+with+Asia&priority=Sustainability&elaborations=true&elaborations=false&scotterms=false&isFirstPageLoad=false [Accessed 11 Mar. 2019].

Felder, R. (2002). Learning and Teaching Styles. [online] Winbev.pbworks.com. Available at: http://winbev.pbworks.com/f/LS-1988.pdf [Accessed 18 Mar. 2019].

Nielson, J. (2012). Usability 101: Introduction to Usability. [online] Nielsen Norman Group. Available at: https://www.nngroup.com/articles/usability-101-introduction-to-usability/ [Accessed 17 Mar. 2019].

Tang, R., Shen, B., Sang, Z., Song, A. and Goodale, M. (2017). Fitts’ Law is modulated by movement history. Psychonomic Bulletin & Review, 25(5), pp.1833-1839.

The Department of Education Tasmania. (2019). Kindergarten in Tasmanian Government Schools. [online] Available at: https://www.education.tas.gov.au/parents-carers/parent-fact-sheets/kindergarten-tasmanian-government-schools/ [Accessed 11 Mar. 2019].

Vu.edu.au. (2019). Oldest VCE student keen for more study | Victoria University | Melbourne Australia. [online] Available at: https://www.vu.edu.au/about-vu/news-events/news/oldest-vce-student-keen-for-more-study [Accessed 18 Mar. 2019].

MAYHEW, D. J. 2006. The Usability Engineering Lifecycle: A Practitioner\'s Handbook for User Interface Design