Application of Software Accessibility Guidelines to Game User Interfaces

Ben Kelly
University of Technology, Sydney
Sydney, Australia
benjamin.kelly@student.uts.edu.au

***Abstract*—This paper explores the unique challenges games and interactive media have in the development of accessible user interface (UI) design patterns, in contrast to the standards and design systems developed and adopted for web and software application development in recent years. A literature review of the current state of both game and software UI accessibility was conducted in order to compare and explore the gaps in the game design ecosystem that software standards have the opportunity to fill. Two primary sources emerged from this analysis, the Web Consortium Accessibility Guidelines (WCAG), the most prolific software accessibility standards, and the Games Accessibility Guidelines (GAG). A synthesis and shortlisting of the most applicable standards from these sources was conducted, with the aim to explore where the WCAG standards could reinforce and provide concrete implementation patterns. The resultant shortlist was then adapted into a set of guiding principles. Highlighting of the challenges involved with the implementation of these principles are explored. Potential implementation frameworks and utility plugins are proposed that could be developed to aid in giving games a baseline of accessible UI to build from. The results of this paper form a starting point of guidelines, recommendations, and learnings for future research and development in the field of accessible design.**

Keywords—accessibility, ui, games, software, web, design, framework, plugins

I. Introduction

Video games, and other forms of interactive media exist in a liminal space between artistic expression and traditional software development, a game must exist both as engaging, creative media while also being a functional piece of software. Because of this, standardised elements and design patterns in software, such as a UI, have a harder time adapting to games where they must also fulfil the goals of being unique, engaging, and expressive to some degree.

Accessibility and standardisation generally go hand in hand, as a common design approach can lead to more room for focusing on the specifics of accessibility, and the adoption of best practices can naturally evolve from the success an increased audience accessibility can bring. While a lot of progress has been made in recent years, games face an uphill battle developing a widely adopted set of accessibility best practices. Mark Brown's 'Designing for Disability' series and yearly review highlights how there has been an increase in many games having individual elements included, yet there is still to be widely adopted standards.[1]

The hypothesis that can be extracted from this is that many of the best practices within software have the potential to be adapted into a common framework that a game's UI could base itself off. The aim of this paper is to synthesise the best practises within the most common software design systems and see where they can apply to games. While there are many facets to accessibility, especially in the implementation of game mechanics, this paper will be focusing primarily on design principals relating exclusively to a games user interface (UI), as the mechanical interactions are less transferable between games and traditional software.

A. Author acknowledgement

As this paper has a focus on accessibility in a broad sense, there will be discussion of various disabilities these design practices are often formed in response to. It is important to note that the author of this paper does not have any significant disabilities that affects how they play games and does not intend to speak on behalf of those with disabilities. It is the authors recommendation that any takeaways from this paper should keep this in mind, and the voice of those most directly affected by accessible design are kept in the forefront of these conversations.

II. Background

A. Current state of games accessibility

From a systematic review of the state of accessibility research in video games, a primary conclusion was that the focus of research has not been on standardised formats, and "Our proposal is to create an integral soft-ware engineering methodology that considers accessibility guidelines, techniques, strategies, human factors, etc. in the video game software development process."[2] General conventions have risen, but there is yet to be a predominant standard, and as such accessible design is still seen to many as a lower priority addition, rather than fundamental aspect. The most robust set of guidelines are the Games Accessibility Guidelines, "A collaborative effort between a group of studios, specialists and academics" [3], with a checklist of guidelines rated by implementation difficulty. These guidelines are a sufficient level of depth to get started in accessible design, but do not emphasise when or how these patterns should be adopted into your game. These guidelines can be used as a foundational element to explore how software standards can be mapped game UI.

A common breakdown of disabilities in respect to categorisation is visual, hearing/auditory, mobility/motor, & cognitive, as represented in the above-mentioned GAG and literature reviews of accessibility. While UI affects all of these to some degree, most relevant UI design patterns are related to visual and cognitive, with the other two having more focus in mechanics and interactions. Many patterns can also solve for multiple of these categories, for example "Ensure no essential information (especially instructions) is conveyed by text, colour, or sounds alone" caters to visual, auditory, and cognitive difficulties, as having multiple ways

to reinforce critical information means there is less chance for not just people with disabilities to miss out, but any general player.

B. Standards in software development

In contrast, software has been standardising many of its practices in relation to User interface and experience (UI/UX), due to a synergy of increased human centred design leading to the development of patterns considered best practice, as well as the reliance on agile implementation and rapid prototyping leading to the increased adoption of sophisticated frameworks such as React or Angular. Material Design has been developed and maintained by Google since 2014 has become the most widely adopted UI system for web and mobile development, with its own design principals both related and independent to accessibility [4]. The Web Consortium (W3C) developed the Web Content Accessibility Guidelines (WCAG), which Material and other design systems accessibility principles are built to adhere to. The co-development of these systems has created an ecosystem where it is more common than not to see software building from these or similar frameworks than to construct their own system from scratch.

Due to the predominance of the WCAG standards in software, it has been selected as the primary source for transferable standard into game UI. Study has been done on the application of these standards in mobile games, and found that a lack of awareness and direct applicability reduce their adoption into games, "Whilst the current Web Accessibility Guidelines can be used as a temporary solution, additional work is required if they are to be used successfully outside of the web domain and into recreational areas such as mobile games."[5] Because of this, the focus of this research will be in synthesising these standards into the most applicable guidelines, and highlighting where and how they can fit in.

III. Approach

A. Building a pattern list

In order to form a set of the most essential guiding principles, the primary sources of the GAG checklist and WCAG standards were filtered and combined in the following process:

1) The GAG has a checklist in the form of a spreadsheet that was filtered by the guidelines' relevance to UI design. This was used as our list of goals we are looking to the WCAG to find related standards and potential implementations for.

2) The WCAG has a quick reference guide that was transposed into a related spreadsheet, with the standards filtered and grouped by both their relevance to UI and immediately discounting standards not applicable to games. Some of the design patterns specified in the standards catered to multiple checklist items and were as such merged, with the related WCAG standards grouped.

3) The resulting lists were merged into a master list, with duplicates between these lists combined/ removed, resulting in a final count of 78 potential design patterns.

B. Selection criteria for shortlisting

While all these design patterns can be considered relevant to a framework, to reduce the complexity and scope, only the most applicable patterns could be focused on. Selection criteria were developed to filter the patterns into a shortlist of results that would best suit a framework:

1) Foundational: Is this pattern required or in service of other accessibility, and is it a required design early in the development process (i.e., it can't just be added on towards the end of a project with ease)?

2) Useful: Does this pattern solve an essential problem that would otherwise cause restrictions of play for those with related impairments?

3) Commonality: Can this design pattern relate to a wide variety of interfaces, and does this apply to multiple genres of games?

4) Implementable: Will a framework or tool aid in the implementation of this pattern? Some patterns, while useful, may be too context specific to allow for a generalised baseline. Luckily the GAG list includes difficulty levels for its patterns which can be used as a starting point for this.

IV. Results

The resulting list of 8 patterns were chosen from this process, and cover a wide variety of UI implementation, from responsive design, readability, interactivity, and comprehensiveness.

TABLE I. Shortlisted Design Patterns

ID	Design Pattern	W3C Guideline(s)
1	Programmatically store sequence of information presented	1.3.1-2, 4.1.1
2	Ensure no essential information (especially instructions) is conveyed by text, colour, or sounds alone	1.3.3
3	Text and images must be against a background of contrast 4.5:1 for small text and 3:1 for large	1.4.3
4	Allow for the dynamic resizing of text up to 200% without functionality loss	1.4.4
5	Pointer Cancelation, actions shouldn't trigger on mouse down and need a reversal/ undo of actions if a user misclicks	2.5.2
6	Ensure a minimum target size for on screen inputs/ interactable items	2.5.5
7	Mechanism for identifying specific definitions & pronunciations for unusual/ context specific words/ abbreviations	3.1.3-4, 3.1.6
8	Consistent navigation mechanisms between all menu screens & identification of common components	3.2.3-4

V. Implementation Challenges

While these are the best suited patterns for a framework, they come with their own set of challenges that must be addressed.

1) A games UI is generally formatted in a more modular fashion as opposed to the linear flow of a website or application. UI elements often situated in the corners or centre of the screen, with what elements are relevant at the

time changing during play. A programmatic layout will need to either be predetermined in a way the designer feels appropriate, or dynamically adapted in real time based on the most important information.

2) Game UI has the goal of being artistically consistent and with a fixed size, so lots of UI is built to use imagery or layouts that would have trouble scaling dynamically. This often results in text being squashed into a fixed size, with a lower than accessible font size being used.

3) Implementing design systems across multiple engines requires extensive rework as every engine utilises its own UI system. This means there isn't the same level or common application of prebuilt UI elements as there is in software. The closest parallel to this would be how Google's Material UI has implementations in most common frameworks for web and app development, such as React [6]. Middleware for UI such as Autodesk Scaleform (discontinued)[7] do exist but have a much lower adoption rate as most are not open source or free.

4) Text and iconography are often positioned against a moving frame, common when a camera moves around the scene, which can result in difficulty having a consistent background contrast, and testing that text is readable in any circumstance. The immediate solution to this would be to always have a contrasting backing to text, a practice adopted by many subtitle systems recently, but this is not always applicable.

It's worth noting that many of these challenges are based off generalities, and there are exceptions in both directions, not all software uses standardised UI such as material, and not all games require non-standard interfaces.

VI. Recommendations

A. Potential Frameworks and Plugins

For the implementation of these guidelines, having a set of utilities and tools that form a cohesive framework would allow a developer to integrate these patterns into their workflow from the very start of development, which is essential for these foundational patterns. The following list is a non-exhaustive list of potential solutions that could be built into said framework.

1) Programmatic storage of navigation flow: A utility that can keep track of the layout flow of your UI as nested information, similar to markup languages. This could be either inbuilt into the UI engine, or implemented as a labelling system (i.e., marking sections within your UI in the editor that can be compiled into nested logic on compilation). This can solve for pattern **#1** and potentially **#8**, as the programmatic flow can be used as a baseline for building consistent navigation structures.

2) Multiple reinforcement techniques: For pattern **#2**, there isn't any specific tooling or utility that can help with this process beyond careful design considerations. It is useful to look at general designs that solve for a similar problem, as this design can be applied not just for impairments such as colour blindness, but also as a general technique to ensure information is digested and reduce cognitive load on any one sense. The writers of Disco Elysium use this technique through their skill systems resulting dialogue, with multiple skills giving you the same core information in different ways, so you are less likely to miss it, as discussed in an interview with the lead designer Robert Kurvitz.[8]

3) Contrast and scale detection: For patterns **#3** and **#4**, A way to keep track of these and pick up on any potential issues early could be inbuilt debug components to test text scaling and contrast in real time. For scaling design, starting with a bottom up approach with looking at how larger scaling will affect your UI first, similar to how responsive UI design often utilises a 'mobile first' approach. Automatic checks and warnings for text contrast can be made by sampling the background of text during play. A similar plugin is available in the design tool Figma to check the WCAG compliance of web designs.[9]

4) Configurable cursor system: Many game engines have quite simple input configurations, checking for mouse/ key/ button inputs frame by frame. This can be built upon to provide a baseline interaction set for accessible cursor inputs, such as clicking, double clicking, and holding down for menu confirmations. This is a design pattern common in virtual cursors for controller inputs, such as in No Mans Sky [10]. For flexibility, these inputs should be remappable to cater to those who have trouble with quick or prolonged input motions, and can cater to pattern **#5**

5) Starter UI kit: Many game engines include a similar starter set of basic buttons and inputs. This however could be expanded upon to provide a more comprehensive set of accessible menu and interaction layouts that could be used as a customisable baseline. This has applicability to help many patterns due to its expandability, but mostly focuses on core interactions covered in patterns **#6** and **#8**.

6) In game glossary: For text heavy interfaces, highlighting of keywords with the ability to hover for explanatory tooltips with a relevant definitions and pronunciations could be built in as a foundational mechanic, covering pattern **#7**.

B. Transferable Learnings

This process has surfaced some important learnings that should be addressed for anyone looking into expanding on this research, or developing any proposed framework utilities:

1) Framework development is difficult to decouple from engine specific implementation, and any future research or development of these tools should start with the scope of a single engine before expanding outwards.

2) Accessibility needs to be taken into consideration as early as possible in the design process, as retrofitting games to be accessible can be limiting. The focus of any frameworks is best served as a starting point for new games.

3) While accessibility is often seen as just accommodations for disability, accessibility guidelines are often benchmarks for general UI principals that make the software more approachable for all users. As an example, programmatic layouts provide benefits both in the application

of technology such as screen readers, but also in the responsive layout of UI for different display aspect ratios, and the dynamic customisation of the UI to a players preference.

C. Future Research Potential

This field has a vast amount of potential for deeper exploration, some interesting topics were uncovered during the research for this paper that would serve well as an expansion of these topics:

1) Comparative study of relevant disabilities against solutions to get a quantitative review of a pattern's effectiveness.

2) The prototyping and testing of implementations of these solutions with players with various accessibility requirements to validate the effectiveness of these proposals.

3) As mentioned above, screen readers are a tool essential to visually impaired players. An in-depth review of using screen reader technology in games, utilising programmatic UI structure and navigation could uncover.

4) Breaking down patterns by genre for more precise patterns. Many useful design patterns were cut due to their applicability to only select genres but could prove fruitful within that niche.

5) This topic covered User interface, but the other aspect of games is commonly the rendering of 3D space. The techniques used to render 3D space can often bias certain cognitive comprehension [12], and a study into accessibility for 3D rendering patterns could be covered in a similar way to this paper.

VII. Conclusion

The goal of making games and software accessible is a never-ending challenge, due to the varying nature of an audience, it is effectively impossible to make a product perfectly accessible, regardless of artistic licence and implementation difficulties. This does not mean there isn't immense value in the striving to make games a more accessible medium of expression, as critical thinking about design choices can not only increase your potential audience reach, but also improve the experience for your existing audience. This paper has shown how best practices in UI can improve the consumption of information across the board and open the potential for more dynamic ways of play if implemented with care and consideration to user experience. The software development industry has a level of maturity and reach that games are working to reach, yet in a similar way to how games have commonly adopted artistic practices from cinema, these lessons can be transferred and adapted to create a unique medium of expression unlike its counterparts.

The results of this paper should be taken as a starting point for a more critical focus into the adaptation and implementation of accessibility guidelines. Many patterns have emerged naturally over the last few years of game developments maturity, some notable examples being subtitles and colour-blind options, and it is in the formalisation efforts that they stop being notable exceptions and start becoming the bar that games must pass to be considered compliant. The final takeaway should be that framework and standards or not, accessibility needs to be core to a game's design principals and kept in mind from the start. Having framework and tools can help with this process, but it is up to the designers and developers to keep it as a priority.

References

[1] Brown, M. and Anderson, S.L., 2020. Designing for Disability: Evaluating the State of Accessibility Design in Video Games. *Games and Culture*, p.1555412020971500.

[2] Aguado-Delgado, J., Gutierrez-Martinez, J.M., Hilera, J.R., de-Marcos, L. and Otón, S., 2020. Accessibility in video games: a systematic review. *Universal Access in the Information Society*, *19*(1), pp.169-193.

[3] Ellis, B., Ford-Williams, G., Graham, L., Grammenos, D., Hamilton, I., Lee, E., Manion, J., Westin, T., 2021, Game Accessibility Guidelines, <http://gameaccessibilityguidelines.com/>

[4] Google 2021, Material Design Accessibility Standards <https://material.io/design/usability/accessibility.html>

[5] Wilson, A. and Crabb, M., 2018. W3C Accessibility guidelines for mobile games. *The Computer Games Journal*, *7*(2), pp.49-61.

[6] Material UI 2021, A Popular React Framework <https://material-ui.com/>

[7] Autodesk 2021, Scaleform <https://www.autodesk.com/products/scaleform>

[8] GameSpot 2021, The Feature That Almost Sank Disco Elysium | Audio Logs <https://www.youtube.com/watch?v=9X0-W5erEXw>

[9] Carr A, Contrast – Figma Plugin, <https://www.figma.com/community/plugin/748533339900865323/Contrast>

[10] Hello Games 2016, No Mans Sky

[11] Boyd D 2014, Is the Oculus rift sexist? <https://qz.com/192874/is-the-oculus-rift-designed-to-be-sexist/>

[12] GameSpot 2021, The Feature That Almost Sank Disco Elysium | Audio Logs <https://www.youtube.com/watch?v=9X0-W5erEXw>