|  |
| --- |
| A Blind Accessible Approach to Game Engine and Tools Development  Bridget A. Casey  BSc Hons Computer Games Technology,  4th Year |

|  |
| --- |
| School of Design and Informatics  Abertay University |

Contents

[Table of Figures ii](#_Toc132385708)

[Table of Tables iii](#_Toc132385709)

[Acknowledgements iv](#_Toc132385710)

[Abstract v](#_Toc132385711)

[Abbreviations, Symbols and Notation vi](#_Toc132385712)

[1. Introduction 1](#_Toc132385713)

[1.1 Blind Accessibility in Games 1](#_Toc132385714)

[1.2 Accessibility of Modern Game Development Tools 1](#_Toc132385715)

[1.3 Research Question 2](#_Toc132385716)

[1.4 Aims and Objectives 3](#_Toc132385717)

[1.5 Hypothesis 3](#_Toc132385718)

[1.6 Overview 3](#_Toc132385719)

[2. Literature Review 3](#_Toc132385720)

[2.1 Blind Accessible Software Design 4](#_Toc132385721)

[2.2 Previous Implementations in Game Engines 4](#_Toc132385722)

[2.2.1 eAdventure 2.0 4](#_Toc132385723)

[2.2.2 Blind Adventure 4](#_Toc132385724)

[2.2.3 ShadowRine Audible Mapper 5](#_Toc132385725)

[2.3 Summary 5](#_Toc132385726)

[3. Methodology 5](#_Toc132385727)

[4. Results 6](#_Toc132385728)

[5. Discussion 7](#_Toc132385729)

[6. Conclusion and Future Work 8](#_Toc132385730)

[References 9](#_Toc132385731)

[Bibliography 10](#_Toc132385732)

[Appendices 11](#_Toc132385733)

# Table of Figures

(To add.)

# Table of Tables

(To add.)

# Acknowledgements

(If any.)

# Abstract

(To write. 300 words. 3.05% of word count. Upper limit approximately 350 words.)

# Abbreviations, Symbols and Notation

(If required.)

# 1. Introduction

For more than two decades, there has been a growing push from game developers and players alike to create more accessible experiences for users with disabilities, including those with limited sight. The most popular games today encompass a wide variety of genres, each with their own style of play, from relaxing farming simulators like Stardew Valley[[1]](#footnote-1), to fast-paced first-person shooters like Doom Eternal[[2]](#footnote-2). However, one common element unites them all – they are heavily reliant on graphics to communicate. But for someone living without sight, this reliance on visuals means key information is always lost.

## 1.1 Blind Accessibility in Games

Historically, accessibility options have fallen by the wayside during the game development process for a multitude of reasons, many of which stem from a lack of knowledge about the subject. A 2022 study led by RNIB found that approximately 20% of studios believe incorporating support for blind players would compromise the quality of their games, while some 64% are unsure how to even accommodate these needs in the first place (IGDA GASIG, 2022). Despite these difficulties, some developers have succeeded. In the early 2000s, studios like Zform went to the extent of creating parallel audio and graphical user interfaces, so that every object and action in their games with a visual component had accompanying audio feedback (Andersen, 2002). More recently, The Last of Us Part II[[3]](#footnote-3) was noted for its bespoke audio narration system, which provided sound cues for all gameplay events and interactions, allowing many visually impaired gamers to play unassisted.

## 1.2 Accessibility of Modern Game Development Tools

Despite studios making great strides towards blind accessibility in their games, the same cannot be easily said for their development tools. Two of the best-known commercial game engines are Unity[[4]](#footnote-4) and Unreal Engine[[5]](#footnote-5). They’re popular among large and small studios alike due to their power and affordability, but they lack many of the same assistive techniques and design choices as the games they’ve been used to create (Tyagi, Choudhary, and Majumdar, 2019). In Unreal Engine 5, there exists limited support to help developers with integrating screen readers into their games, but not for use within the engine itself. Almost no element of the user interface is compatible with screen readers, as text and buttons are rendered as plain images.

Additionally, in both Unity and Unreal, the default graphical interface can easily become cluttered with widgets. This means that even if a screen reader could be used, it would likely struggle to interpret relevant information in a coherent manner.

One might assume a visual impairment would prevent a person from thriving in a software development role, but approximately 1 in every 100 programmers is blind (Larson, Q., 2017). This is partly due to the increased availability and affordability of assistive technologies like the screen reader, which has existed for decades as the primary tool for helping people who are visually impaired interact with computer software. Current popular screen readers, such as the free and open-source NonVisual Desktop Access (NVDA)[[6]](#footnote-6), can interact with busy webpages and text-based desktop applications, providing audio feedback when the environment changes.

Many modern programming IDEs, such as Visual Studio[[7]](#footnote-7) and Eclipse[[8]](#footnote-8), are also used successfully by software developers who rely on screen readers. Just within the past few years, Microsoft has integrated support features into Visual Studio specifically for this purpose, such as the ability to navigate the entire display using only keyboard controls (Coding Tech, 2018).

## 1.3 Research Question

If a game developer who relies on screen reading software desires to use certain industry standard tools, create graphical applications, or fill a role other than programming, their options are limited. A screen reader can do nothing when presented with plain icons or complex geometry. In these cases, the only alternative is to build a custom engine to suit their needs, complete with bespoke audio and descriptions (Hamilton, 2015).

However, this is not a realistic solution for many. A game engine is often an enormous application with an extensive list of features and sub systems, which can take a great number of people and a long time to develop. This results in the engine’s editor having a complex graphical user interface, far more than the average text document or website, rendering a screen reader unusable.

“To what extent can a graphical game development tool be made blind-accessible using custom audio feedback and minimalist UI design?”

## 1.4 Aims and Objectives

The overarching goal is to develop a bespoke level editor tool for simple 2D games, which embodies common design principles for blind accessible software. More specific objectives are to:

* Implement underlying game engine architecture based on existing, open-source frameworks, which contains core elements such as a 2D renderer, input event system, entity component system, and audio manager.
* Integrate support for compatibility between UI elements and third-party screen reader applications, using existing libraries.
* Implement a bespoke GUI-based editor using chosen design techniques, built upon the custom engine framework.
* Evaluate the effectiveness of the solution through qualitative testing with a small group of visually impaired users in the target audience.
* Revise prototype design based on feedback from test participants.

## 1.5 Hypothesis

The implementation of multiple techniques commonly used to accommodate visually impaired users in other types of software can be applied to game engine editors. These include screen reader compatibility, simplified menu navigation, event-based audio cues, and large, high-contrast text. It is hypothesised the combined use of several smaller features will allow sightless users to navigate the prototype tool and use it to create levels for a simple game without assistance from a sighted person.

## 1.6 Overview

The Literature Review will explore previous examples of where blind accessibility in game development tools has been achieved. The Methodology will provide a detailed breakdown of the steps undertaken to design and develop the level editor prototype. The Results chapter will detail the findings from the user testing sessions conducted with visually impaired participants. The Discussion chapter will evaluate these results in the context of the prototype. The Conclusion will explain the implications of this research in relation to large-scale game engine design, as well as how the chosen implementation could be revised or extended to better meet the needs of its userbase.

# 2. Literature Review

(To be finished. 1700 to 1850 words average. 18.78% of word count. Upper limit approximately 2250 words.)

## 2.1 Blind Accessible Software Design

According to recent research, creating blind-accessible game development tools is possible, but it requires careful consideration when designing the interface (Tyagi, Choudhary, and Majumdar, 2019).

Developers should aim to reduce the number of user inputs required to achieve an outcome. They should also ensure at least two forms of feedback, such as visual, auditory, and tactile, exist for every action.

## 2.2 Previous Implementations in Game Engines

### 2.2.1 eAdventure 2.0

The eAdventure tool (Torrente et al., 2014) is an open-source application intended for creating simple 2D, point-and-click style games for web browsers. It was developed in Java and initially released in 2005.

Beginning with version 2.0, an accessibility module was integrated to allow for various support features, including the use of screen readers. This enabled the user to exclusively use the keyboard to navigate the application. Additionally, the Accessibility Core ensured game objects were compatible with screen readers by changing how they were rendered. Object descriptions were displayed as text and then fed through a text-to-speech API bundled with the application. Other effects also existed, such as the ability to generate a textual description of the environment’s contents when entering a new scene.

### 2.2.2 Blind Adventure

Blind Adventure (Stadler and Hlavacs, 2018) is a tool for creating simple, exclusively audio-based games for iOS and Android devices. It was unique in that it was aimed primarily at blind game developers. Upon loading the app, users would select between two modes – play a previously created game or make a new one.

In engine mode, the application employed various design choices aimed at improving the experience of sight-impaired users. These included large fonts and high contrast, as well as spoken audio cues when a scene changed or when a choice was presented on-screen. Additionally, navigation was simplified to a handful of swipe gestures. Consideration was given to how choices could be presented simply, and this was achieved using a node-based system. This meant no more than two buttons were displayed at one time. e.g., ‘add level’ and ‘edit level’. Orientation was also locked to portrait mode, as a blind user would have no way of telling if this changed.

Based on feedback gathered by the researchers, experience with the app was positive overall, except for minor issues related to the speed of swiping during menu navigation.

### 2.2.3 ShadowRine Audible Mapper

ShadowRine is a 2D Japanese action roleplaying game in which the player can explore a world and fight monsters across many maps. The developers later released a graphical editor tool for players to create custom maps. However, the ShadowRine Audible Mapper (Matsuo et al., 2016) is different in that it can be fully navigated by blind users.

The developers chose Hot Soup Processor (HSP) as their programming language, as this allowed for significant screen reader support. For ease of use, the mapper tool permitted users to move the mouse cursor using keyboard arrows and select objects using number keys. For speed purposes, users could also select a range of scene objects and work on them all at once. Audio feedback also featured dynamic sound effects. A cue would play to indicate where the mouse cursor was, with varying pressure depending on its distance across each axis.

Based on the researchers’ testing, sight-impaired users were able to create maps effectively using this tool, although navigation sometimes took trial and error.

Some users complained that the sound system was confusing, and the maps were too large, leading them to get lost when placing objects.

## 2.3 Summary

This chapter gave an overview of commonly used techniques for creating blind accessible software and discussed examples where they have been applied in the context of game engines and editor tools. The Methodology chapter will explore how these techniques, as well as others, have been adapted for use in designing and developing the prototype application.

# 3. Methodology

(To write. 2750 to 3000 words average. 30.46% of word count. Upper limit approximately 3750 words.)

<Chapter plan here.>

# 4. Results

(To write. 500 to 800 words. 8.12% of word count. Upper limit approximately 1000 words.)

<State results of user testing sessions.>

# 5. Discussion

(To write. 2250 to 2500 words. 25.38% of word count. Upper limit approximately 3000 words.)

<Evaluate results of user testing sessions. What features worked, what didn’t, and why?>

# 6. Conclusion and Future Work

(To write. 600 words. 6.09% of word count. Upper limit approximately 750 words.)

<Chapter plan here.>

# References

Andersen, G. (2002) *Playing by Ear: Using Audio to Create Blind-Accessible Games*. Available at: https://www.gamedeveloper.com/audio/playing-by-ear-using-audio-to-create-blind-accessible-games (Accessed: 5th October 2022).

Coding Tech. (2018) *How A Blind Developer Uses Visual Studio*. Available at: https://www.youtube.com/watch?v=94swlF55tVc (Accessed: 7th October 2022).

Hamilton, I. (2015) *Screenreaders and game engines*. Available at: http://ian-hamilton.com/screenreaders-and-game-engines/ (Accessed: 28th September 2022).

IGDA GASIG (2022) *RNIB Accessible Gaming Research 2022*. Available at: https://www.youtube.com/watch?v=aOzglVUb61c (Accessed: 1st March 2023).

Matsuo, M., Miura, T., Sakajiri, M., Onishi, J., Ono, T. (2016) ‘Audible Mapper & ShadowRine: Development of Map Editor Using Only Sound in Accessible Game for Blind Users, and Accessible Action RPG for Visually Impaired Gamers’, Computers Helping People with Special Needs. ICCHP 2016. *Lecture Notes in Computer Science*, vol. 9758. Springer, Cham. DOI: 10.1007/978-3-319-41264-1\_73

Stadler, V., Hlavacs, H., (2018) ‘Blind Adventure - A Game Engine for Blind Game Designers’, *CHI PLAY '18: Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pp503-509. Association for Computing Machinery. DOI: 10.1145/3242671.3242703

Torrente, J., Serrano-Laguna, Á., Aguado, Á., Moreno Ger, P., Fernández-Manjón, B., (2014) ‘Development of a Game Engine for Accessible Web-Based Games’, Games Learning Alliance, GALA 2013. *Lecture Notes in Computer Science*, vol. 8605, pp107-115. Springer, Cham. DOI: 10.1007/978-3-319-12157-4\_9

Tygai, M., Choudhary, C., Majumdar, R. (2019) ‘A Study on Gaming Engines Accessibility’, Advances in Interdisciplinary Engineering, pp33-41. *Lecture Notes in Mechanical Engineering*. Springer, Singapore. DOI: 10.1007/978-981-13-6577-5\_4

Larson, Q. (2017) *What I learned from Stack Overflow’s massive survey of 64,000 developers*. Available at: https://www.freecodecamp.org/news/73-of-programmers-are-web-developers-and-other-insights-from-stack-overflows-massive-2017-survey-89a13b11e370/ (Accessed: 3rd March 2023).

# Bibliography

Sightless Kombat (2020) *The Last of Us 2 – Blind Accessibility Review*. Available at: https://caniplaythat.com/2020/06/18/the-last-of-us-2-review-blind-accessibility/ (Accessed 1st March 2023).

Wilds, S. (2020) *For All The Players: A History Of Accessibility In Video Games.* Available at: https://www.gamingbible.com/features/games-for-all-the-players-a-history-of-accessibility-in-video-games-20200124 (Accessed: 28th February 2023).

# Appendices

(To add.)

1. ConcernedApe (2016) *Stardew Valley* [Video game]. ConcernedApe. [↑](#footnote-ref-1)
2. id Software (2020) *Doom Eternal* [Video game]. Bethesda Softworks. [↑](#footnote-ref-2)
3. Naughty Dog (2020) *The Last of Us Part II* [Video game]. Sony Interactive Entertainment. [↑](#footnote-ref-3)
4. Unity Technologies (2022) *Unity* (Version 2022.1.18) [Computer program]. Available at: <https://unity3d.com/> (Accessed: 8th October 2022). [↑](#footnote-ref-4)
5. Epic Games (2022) *Unreal Engine* (Version 5.0.3) [Computer program]. Available at: <https://www.unrealengine.com/> (Accessed: 8th October 2022). [↑](#footnote-ref-5)
6. NV Access (2022) *NVDA* (Version 2022.3.1) [Computer program]. Available at: <https://www.nvaccess.org/> (Accessed: 7th October 2022). [↑](#footnote-ref-6)
7. Microsoft (2022) *Visual Studio* (Version 2022 17.3) [Computer program]. Available at: <https://visualstudio.microsoft.com/> (Accessed: 8th October 2022). [↑](#footnote-ref-7)
8. Eclipse Foundation (2022) *Eclipse IDE* (Version 2022-09) [Computer program]. Available at: <https://www.eclipse.org/> (Accessed: 8th October 2022). [↑](#footnote-ref-8)