# UnityAccessibilityToolkit (UA11Y): Developer Tool and Roadmap to Make Games More Accessible for People with Vision Impairments

Klemens Strasser
Graz University of Technology
Austria

Johanna Pirker
Graz University of Technology, Austria &
Ludwig-Maximilians-Universität, Germany &
jpirker@tugraz.at

## Abstract

*An increasing number of game productions rely on accessibility features. Large productions like The Last of Us demonstrate what is possible in this area, but many games still lack essential accessibility features. Smaller productions often lack the knowledge and resources to implement different accessibility features.*

*In the first part of this paper, we investigate how people with vision impairments play video games and compare current accessibility features. Based on our findings, we present a roadmap and guidelines for improving game accessibility for individuals with a color vision deficiency, low vision, or any form of vision impairment.*

*In the third part, we introduce the UnityAccessibilityToolkit, a toolkit for game developers to quickly and easily integrate accessibility elements for players with vision impairments into a game. We evaluated the toolkit with nine developers who were challenged to make a simple match-3 game more accessible. Our results demonstrate that the Unity toolkit is easy and fast to use, and that important accessibility features can be implemented quickly.*

## 1. Introduction

Video games have become an integral part of our lives and serve as a means of social interaction and escapism from daily challenges and difficulties (Calleja, 2010). With online multiplayer games, people can connect with family, friends, and strangers to solve puzzles, quests, and challenges together (AbleGamers, 2020). However, mainstream games mostly rely on the visual abilities of their users, thereby excluding people with visual impairments (Porter, 2014). According to the World Health Organization, an estimated 253 million people worldwide live with a visual impairment, making accessibility a critical issue in game development.

While some games have shown the potential of including accessibility features for this user group, such features are still rare in mainstream game development (Atkinson et al., 2006). In this paper, we aim to explore the gameplay strategies of people with visual impairments and discuss the basic interaction forms and vision accessibility features required for video games.

Based on our research findings, we describe a roadmap and guidelines for areas of improvement to make games more accessible for individuals with color vision deficiency, low vision, or any form of vision impairment. Additionally, we have designed a prototype using the UnityAccessibilityToolkit to help game developers more easily incorporate accessibility features, such as improving navigation in a game world and providing better access to UI elements. We aim to evaluate the effectiveness of this prototype in facilitating the integration of accessibility features by game developers and their willingness to invest time in enhancing their games with such features. This work is based on the thesis by the first author and the full thesis includes additional implementation details Strasser, 2021.

## 2. Related Work

Accessibility in games is an increasingly relevant topic and is more and more also addressed in the AAA industry. We can see many new devices and a much more general awareness of accessible methods, including special controllers, accessibility guidelines, dedicated modes and adjustable controls, and difficult systems. However, creating video games is already a difficult endeavor and comes with so many challenges. Considering accessibility is often an issue disregarded

URI: https://hdl.handle.net/10125/106703
978-0-9981331-7-1

by developers, especially indie developers. However, considering accessibility can give access to more players. So considering accessibility in the game development process is not only important for the the inclusion of players, but also for game developers.
Several strategies have been implemented in the past to help visually impaired users navigate and interact with computing devices. These include features and assistive technologies such as adjusting the graphical user interface, using refreshable braille displays, magnifiers or zoom features, speech synthesizers, and screen readers (Taylor, 2009; Thatcher, 1994). These technologies have made it possible for visually impaired people to use many common applications and programs. However, gaming with a visual impairment can be an added challenge, as many games rely heavily on visual elements.

## 2.1. Games for People with Visual Impairment

The game workflow presents two major challenges for accessibility: not only must the game itself be made accessible, but also the related navigation and title screens (Game Accessibility Guidelines, 2019). Most games are built with cross-platform game engines like Unity[1], which present accessibility challenges of their own.

One issue is that these engines often render the user interface in a way that its elements are not exposed to the screen reader being used, and the in-game menu navigation is not supported by the system screen reader (Andrade et al., 2019). Some games have attempted to make the menu accessible by imitating screen reader-like behavior, where every menu element is communicated to the user through the spoken word (RARECSM, 2019). However, these settings are not always turned on by default, and visually impaired users must find a way to enable them.

We have now established a knowledge of how a visually impaired user can go from wanting to play a game to the actual point of playing the game. While researching games that can be played by visually impaired gamers, we found four different categories of games:

- I. Games that are not accessible at all,
- II. Games that were not intended to be accessible, but the visually impaired community found ways to play them nonetheless. Thus, these games can be called Unintentionally Accessible Experiences. Examples: Madden NFL EA Tiburon, 2020's sports commentary ); Mortal Kombat also features a limited directional movement NetherRealm Studios, 2019 (see Andrade et al., 2019; Vice News, 2019)
- III. Games that communicate the content first and foremost via audio instead of visuals. These games are called Audio Games. Examples: The Blindfold games by ObjectiveEd[2] show how these earcons (auditory icons, which are audio cues or sounds to convey information or provide feedback) can effectively be used to turn well-known games into audio games. Blind Legend is an action-adventure game by DOWiNO (DOWINO, 2016).
- IV. Games that heavily rely on visuals, but have either special accessibility settings or support tools like screen readers to make them accessible to visually impaired players. We call them Intentionally Accessible Mainstream Experiences. Examples: Shredder Chess (Skizzix, 2009), Subwords (Strasser, 2018), (Naughty Dog, 2020).

In addition to mainstream games and games designed for entertainment, there are also games that have been specifically built for scientific studies but were never made available outside those studies. Examples include Blind Hero (Yuan and folmer eelke, 2008) and Rock Vibe (Allman et al., 2009), which are imitations of popular music games Guitar Hero (Harmonix, 2007a) and Rock Band (Harmonix, 2007b). These games require special hardware that was only built for the study and were not released for public use.

## 2.2. Game Development for Vision Accessibility

Today, game development is often done using game engines such as Unity or Unreal, which are designed to make it easier for developers to create games. Therefore, including accessibility features directly into the game engine can be an important and crucial step in making it easier for developers to create accessible games and experiences. In this study, we examined the most popular game engines and compared the available vision accessibility features.

In this study, we focused on the game engine Unity and examined asset packages for vision accessibility for Unity. Asset packages are extensions and plugins for the Unity game engine that provide different features to support game developers. Some of them are even free to use. We found the following packages: The

[1] https://unity.com/

[2] https://blindfoldgames.org

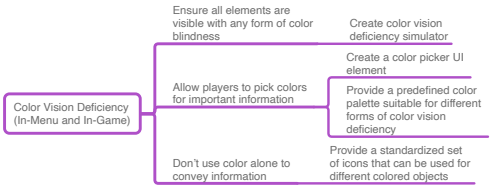


Figure 1: Areas of improvement for individuals with a color vision deficiency

Colorblind Effect[3], which addresses colorblindness; UI Accessibility Plugin[4] (UAP), which makes UI and UI-heavy games accessible by providing a screen reader; and the Responsive Spatial Audio for Immersive Gaming[5], which provides spatial audio.

While some tools are already available, many of them only support specific or limited features. Based on our findings, in the next chapter, we present a roadmap for developments in vision accessibility in the game development process.

## 3. Roadmap for Vision Accessibility

Guided by our discussion of available features and how people with vision disabilities interact with computing devices and games, we have compiled the following potential roadmap to raise awareness of areas where vision accessibility can be enhanced for games created in game engines. Figures 1, 2, and 3 provide an overview of different areas for improvement categorized into three groups: (1) enhancements for individuals with color vision deficiency, (2) enhancements for people with low vision, and (3) enhancements for any form of vision impairment. We distinguish between enhancements that may be added with an extension for the game engine and those (noted with a dashed line) that must be specifically created as part of the game (e.g., adding realistic sound design). These figures provide a first overview of potential areas for improvement but are not intended to be comprehensive.

## 4. Unity Accessibility Toolkit (UA11Y)

Based on our study of related work and the roadmap we developed, we designed the UnityAccessibilityToolkit (UA11Y), a toolkit implemented as a Unity asset package that helps game developers create or update games with accessibility features for visually impaired gamers. The toolkit

[3] shaders/fullscreen-camera-effects/colorblind-effect-76360
[4] gui/ui-accessibility-plugin-uap-87935
[5] game-toolkits/responsive-spatial-audio-for-immersive-gaming-a-microsoft-garage-144702

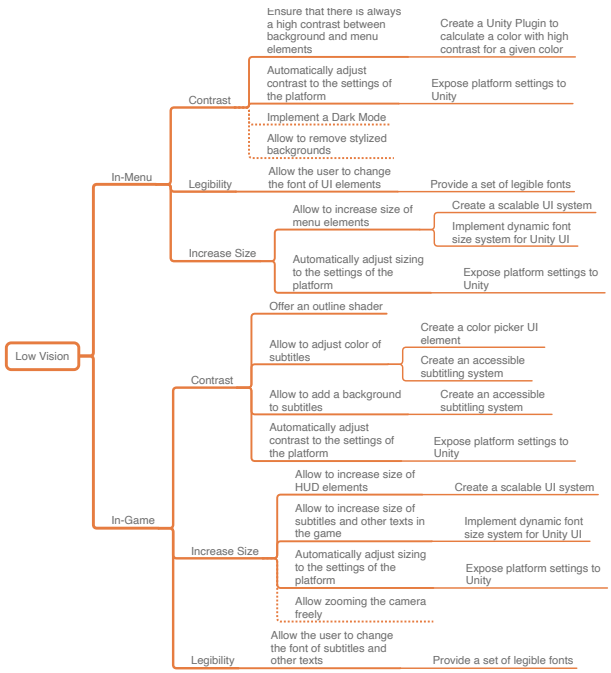


Figure 2: Areas of improvement for individuals with low vision. Dashed lines refers to elements that need to be created as part of the game.

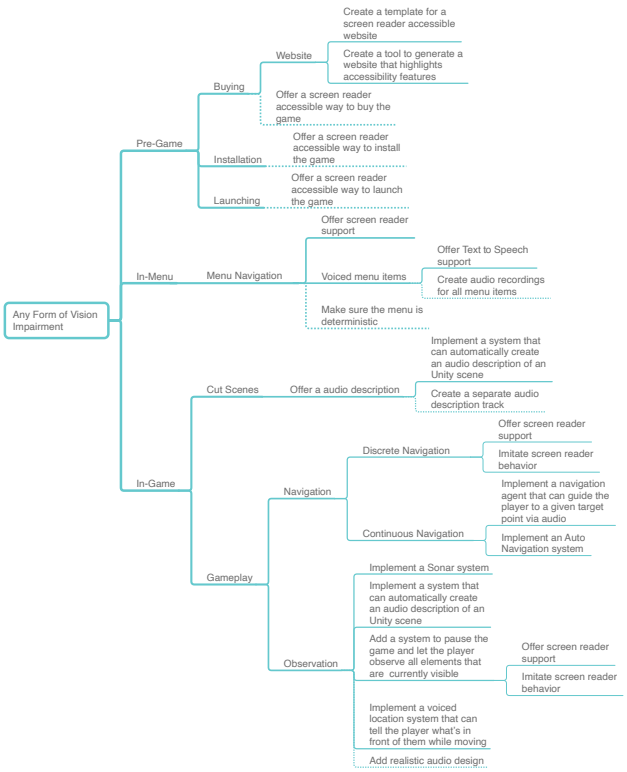


Figure 3: Areas of improvement for individuals with any form of vision impairment

is designed to be easily extended with additional accessibility features. In the first iteration, we focused on improvements for vision accessibility, with a special focus on the features we identified in our roadmap in the previous section:

- *Menu Navigation*: Help navigation through the menu of a game to start the actual gameplay or to adjust settings
- *Environment Observation*: Pausing the game and letting the user observe all the elements that are currently visible
- *Discrete Navigation*: Navigating elements or a character on a fixed grid
- *Continuous Navigation*: Freely moving a character around an environment

Menu navigation, discrete navigation, and environment observation can be realized with a screen reader. Thus, the first part of the toolkit is the implementation of a screen reader for Unity. The second part can be realized with a Navigation Agent.

### 4.1. Accessiblity Signifier

As a basis for accessibility enhancements, we designed and implemented an auditory accessibility signifier that developers can attach to Unity game elements. In his book *The Design of Everyday Things*, Norman, 2002 describes signifiers as *"any mark or sound, any perceivable indicator that communicates appropriate behavior to a person"*. The signifier stores details, including possible interactions, about a game object and adds information on whether a game object is accessible at all. Our accessible signifier is inspired by the `UIAccessibility` protocol from UIKit, a standard UI framework for iOS, and has the following four properties:

- *Label*: A short but concise label of the object.
- *Traits*: Indication of how the game object behaves or should be treated.
- *Value*: The value of the game object.
- *Description*: Detailed description of the function of the game object. This should give additional context to the function of the object and how to interact with it if this isn't obvious through labels, values, and traits.

Figure 4 provides an overview of how this can be used with a button or a slider.

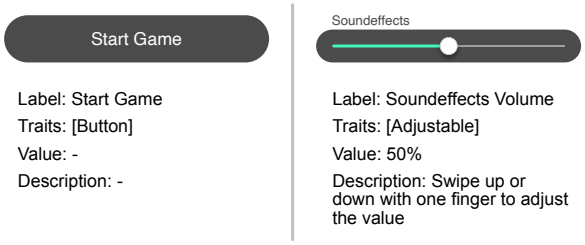


Figure 4: Examples of how to use the values of the accessibility signifier with a button (left) and a slider (right)

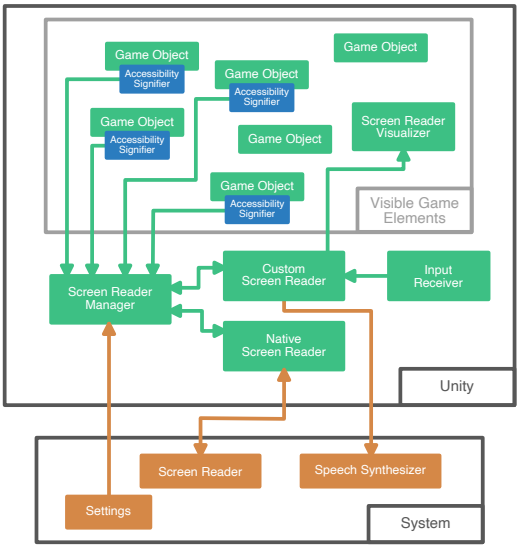


Figure 5: Architecture of the screen reader

### 4.2. Implementation of the Screen Reader

As mentioned earlier, screen readers are one of the most important features for making a game visually accessible. However, game objects in Unity are not visible to other screen readers, such as VoiceOver or Windows Narrator. These screen readers are already well-known and used by users, so it is preferable to use them rather than implementing a new one. Therefore, we implemented a ***Screen Reader Manager*** to capture all accessible in-game objects on the screen. This information is then forwarded to the native screen reader, which is built from scratch for Unity. Figure 5 provides an overview of our architecture. Figure 6 illustrates how the screen reader sorts game objects with the accessibility signifier based on their screen coordinates, which are then sent as an auditory announcement using the system's known speech synthesizer.

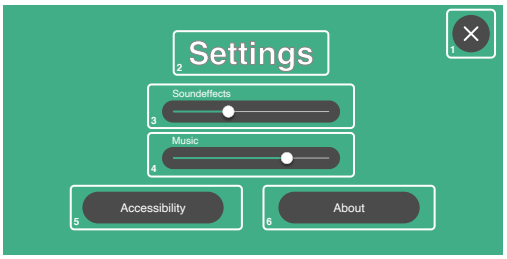


Figure 6: Example of how our screen reader will sort the game object with the accessibility signifier

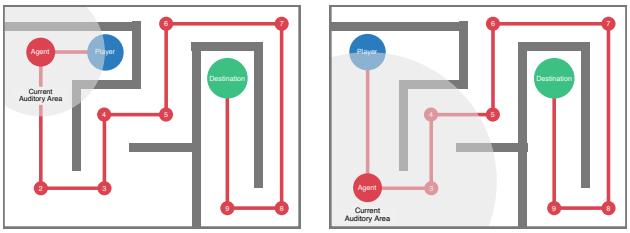


Figure 7: Sketch of how continuous navigation is eased via the navigation agent. Left: Initial position of player and navigation agent. Right: The first intersection was reached and the agent moved to the second one.

### 4.3. Implementation of the Navigation Agent

The navigation agent is designed to enable continuous navigation from any location to other reachable in-game points. The developer has to add a user interface enabling choosing a location, which is sent to our navigation agent. The navigation agent calculates a navigation path and navigates the players through calculated straight lines from point to point through positional earcons. Figure 7 illustrates an example of how this navigation is designed.

### 4.4. Use Cases

In this section, we describe examples of how the implemented features of the toolkit can be used to make games more accessible for visually impaired players.

### 4.5. Menu Navigation

For this use case, we used the FPS Microgame[6] provided by Unity. We added our Game Acccessibility components (UA11YSlider, UA11YTooglet,...) to the UI elements and added our `UA11YScreenReaderManager` prefab to the scene to create the screen reader, finding the accessible objects in the scene, and handing them to the screen

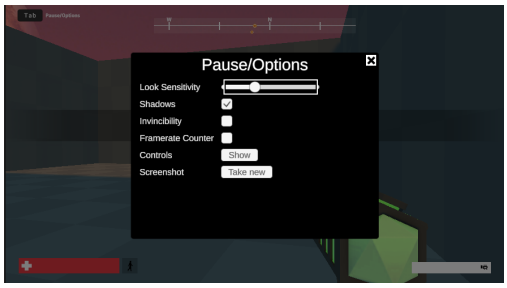


Figure 8: Menu navigation in the Pause/Options menu of the FPS Microgame. The highlighted element is the Look Sensitivity slider.

reader. Figure 8 demonstrates how the menu looks with the activated screen reader.

### 4.6. Environment Observation

Environment observation allows players to pause the game and observe the surroundings via the screen reader (e.g. by stepping through visible objects or browsing the objects with the mouse cursor). We again used Unity's FPS Microgame as a basis.

We utilized the `UA11YElement Component` to label important objects like enemies or door markers and ensured that each element had a corresponding `Collider Component` attached. This was essential for excluding non-visible objects using raycasting techniques to determine intersections between rays and objects[7]. To enable pausing the game with a key press, activating the `UA11YScreenReaderManager`, and calling its `VisibleElementsDidChange()`, we could then navigate through the accessible elements on the screen, as depicted in Figure 9.

### 4.7. Continuous Navigation

Continuous navigation refers to unrestricted movement within a 2D or 3D space, such as exploring open-world environments or locating non-player characters. To make this type of navigation accessible, we developed a navigation agent. We showcased this agent's functionality using the FPS Microgame.

In this game, there is a main enemy called the turret, and we aimed to provide auditory guidance to this enemy using the navigation agent. To achieve this, we simply added the `UA11YNavAgentManager` to the scene and, upon a key press, initiated the navigation guide to the enemy turret `GameObject`. This action triggered the calculation of a navigation path from the player's position to the turret and initiated audio

[6]https://assetstore.unity.com/packages/templates/fps-microgame-156015

[7]https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

(a) Door indicator

(b) Enemy robot

Figure 9: Getting information about the visible objects via environment observation.

navigation along this path. You can view the path in Figure 10.

### 4.8. Discrete Navigation

We also wanted to cover discrete navigation on a fixed grid with the screen reader, but since the FPS Microgame uses Continuous Navigation, we used a different sample project for demonstration. We selected a match-3 puzzle game from a RayWenderlich tutorial[8], similar to Bejeweled **game:Bejeweled**.

To enhance gameplay for individuals with vision impairments, we attached a `UA11YElement Component` to each game tile and labeled them by type. We also provided the grid locations for sequential navigation, although browsing remains a more efficient way to play. Additionally, we subscribed to the `Click UA11Y-Element-Interaction-Event-Type` to activate the tile selection. Lastly, we kept the screen reader updated by calling `UA11YScreenReaderManager VisibleElementsDidChange()` whenever a match occurred. This made the game board fully accessible and interactive through the screen reader, as shown in Figure 11.

[8]https://www.raywenderlich.com/673-how-to-make-a-match-3-game-in-unity

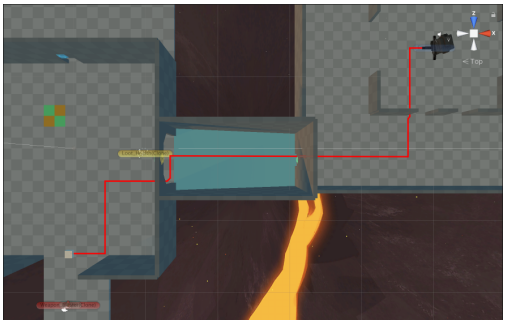

Figure 10: The navigation path calculated by our navigation agent, going from the players' position to the turret enemy.

## 5. Evaluation

To evaluate the usability of the toolkit, we set up an experiment to give Unity developers the task of integrating accessibility features to a given game with our toolkit. The tasks were related to adding the basic accesiblity features to a simple match-3 puzzle game with the screen reader of our toolkit. We focused on developers' feedback, usability measures, and the time they needed for the edit.

### 5.1. Method

We recruited 9 (P1-P9, 1 female, 8 male) game developers between the ages 21-30, most of them with a University background in software engineering via a game-specific Discord server. They were compensated with a 10€ Steam voucher, and no time limit was set. They had different levels of experience with Unity and only one of the used Unity professionally. Table 1 gives an overview of the participants. The data was collected with a pre- and postquestionnaire. This included 5 demographic questions, followed by 10 questions about sample project accessibility with screen readers, and concluded with 11 questions about vision accessibility. Communication was handled through Discord. The participants recorded their screens during the study time. Facilitators were available for questions but did not intervene or help with the tasks. We sent them the same sample match-3 puzzle game as described in the previous section, taken from a RayWenderlich tutorial[9] which consists of three screens (see Figure 11). Figure 11 gives an overview. We sent the participants the Unity project already including the UA11Y toolkit and an overview of the toolkit with a focus on the screen reader part and a task description to enable the screen reading feature for this game as described earlier. We did not intervene or help but

[9]www.raywenderlich.com/673-how-to-make-a-match-3-game-in-unity


(a) Start screen


(b) Game screen


(c) Result screen

Figure 11: The three screens of the match-3 sample game.

were available through Discord in case of problems. In the end, we asked participants to send us their results and fill out a post-questionnaire consisting of questions on a Likert scale between 1 (strongly agree) and 5 (strongly disagree), and open-ended questions. including demographic questions, questions regarding their process, and questions about vision accessibility in general.

## 5.2. Findings

**Accessibility Experience** We also wanted to understand better the general participants' experience with accessibility in general. The results were disillusioning and showed that only 3 out of the participants worked on implementing accessibility features prior to this experiment (color blindness post-processing filters and button remapping for a Unity game). None of them implemented a screen reader feature before. Six out of nine mentioned that they did not think of this possibility, three mentioned that they did not include it because they thought the target group was too small and two mentioned that they wanted to add it but did not find the right tools or features to quickly add it. One participant mentioned, that someone else in their development project added such a feature. In terms of general knowledge about the topic, 7/9 mentioned that they informed themselves about game accessibility in the past (7 watched videos, 6 attended talks, 6 read articles, 1 learned about it during a lecture). 4/9 mentioned that they have used any kind of accessibility features themselves while playing games in the past.

**Accessibility of the Game Results**

Out of the nine participants, six took 90-120 minutes to complete the task, one finished in 60-90 minutes, and two took 150-180 minutes. All participants found the task easy to understand (AVG: 4.44 STD: 0.53) and easily located everything they needed in the package (AVG: 4.78 STD: 0.44). On average, they rated implementing screen reader support as easy (AVG: 4.00 STD: 1.00), with most emphasizing the importance of offering accessibility features in a game (AVG: 4.67 STD: 0.71). The majority expressed willingness to make their next game accessible with a screen reader (AVG: 4.33 STD: 0.50) and utilize the toolkit (AVG: 4.67 STD: 0.50).

When participants were asked about what they found easy to implement, all mentioned making Unity UI elements like text and buttons accessible. One participant expressed, *"All UI elements and text were basically only drag and drop in Unity, which was nice."* This ease of implementation was evident in our assessment, as all solutions made the UI elements on the start screen, result screen, and game screen visible and interactive with the screen reader.

Our assessment highlights the most significant challenge participants faced, which was making the tiles and gameplay accessible. One participant noted, *"The tile was the only thing where no finished script was provided, hence this took the most effort."* Before discussing participants' solutions, it's important to discuss the reference implementation. In the reference implementation, game tiles were made discoverable for both sequential navigation and browsing. Each tile's accessibility description included its type, item location (row and column), and a hint if it was selected. Selection in the reference implementation could be toggled using the standard screen reader selection mechanism.

Among the participants' solutions, almost all differed from the reference implementation, except for one (P8) that matched both in interaction and tile description. However, one participant (P3) missed updating tile descriptions after interacting with the game board, rendering their implementation unplayable for individuals with visual impairments due to a divergence

| Participant | Age Group | Programming Experience | Unity Experience | Uses Unity Professional |
|---|---|---|---|---|
| P1 | 21-30 | 5-6 years | less than 1 year | no |
| P2 | 21-30 | 5-6 years | 3-4 years | no |
| P3 | 21-30 | 3-4 years | 1-2 years | no |
| P4 | 21-30 | 5-6 years | 1-2 years | no |
| P5 | 21-30 | 6+ years | 3-4 years | no |
| P6 | 21-30 | 6+ years | 1-2 years | no |
| P7 | 31-40 | 6+ years | 5-6 years | yes |
| P8 | 21-30 | 6+ years | 1-2 years | no |
| P9 | 31-40 | 1-2 years | 3-4 years | no |

Table 1: Demography of the evaluation participants

in game state.

The remaining seven solutions diverged from the reference but remained playable with closed eyes, as per the participants' requirements. However, there were common issues: (1) Lack of Tile Location: Seven out of nine solutions did not include tile locations, making sequential navigation challenging. (2) Selection State Ambiguity: In six out of nine solutions, it was difficult to determine the selection state of a tile solely by focusing on it with the screen reader. The selection state could only be discerned by attempting to select the tile and listening for an audio cue. (3) Non-standard Selection Mechanism: Four out of nine solutions implemented tile selection via a mouse click, deviating from the screen reader-based interaction in the reference implementation.

Where exactly the solution of each participant differed from the reference can be seen in Table 2.

Limitations of the study can be seen in the small sample size and the self-reporting of the questionnaires. Also, it would be important to enlarge the study to a more diverse set of test users.

## 6. Discussion and Conclusion

In this paper, we presented a roadmap and guidelines to improve the accessibility of games for individuals with vision impairments, based on the study of relevant tools and background. We also developed the UnityAccessibilityToolkit, a tool to help game developers build visually accessible games with the Unity game engine. Our evaluation showed that toolkits can be an effective tool to make it easier to create accessible games and that developers are open to adding different accessibility features to their games if toolkits or guidelines are available.

However, we found that many developers are simply not aware of the necessity and possibility of adding such features, which is a major open issue. Therefore, future research should focus on two aspects: developing more easy-to-use toolkits to make it as easy as possible for developers to add accessibility features and increasing awareness among developers about the necessity and possibility of adding such features.

Regarding the UnityAccessibilityToolkit, future work can include extending its capabilities to cover other accessibility standards for games and creating new tools. Additionally, increasing the appeal for developers to make accessibility more relevant and visible to them can further motivate them to create more accessible games, especially indie game developers.

In summary, our research contributes to the advancement of game accessibility by providing a roadmap and guidelines for improvement and by developing a useful toolkit for game developers. While such a toolkit cannot make every game accessible and probably not every kind of game is originally designed or suitable for such as support, as many games rely on small hints, visual atmospheres, or other cues, a major contribution of this work should be to make it easier for developers to try their best and especially increase awareness. Further work should certainly experiment more also with other elements than just screen reading, such as haptic cues or other feedback. Summarizing, we hope that this work will inspire more research and development in this area and lead to the creation of more accessible games for everyone to enjoy.

| Solution | Visible to Screen Reader | Desc. Includes Type | Desc. Includes Location | Desc. Includes Selection | Desc. Updates | Interactive through Screen Reader | Interactive through Mouse | Overall Playable |
|---|---|---|---|---|---|---|---|---|
| Reference | yes | yes | yes | yes | yes | yes | no | yes |
| P1 | yes | yes | no | no | yes | no | yes | yes |
| P2 | yes | yes | no | yes | yes | | | |
| P3 | yes | yes | yes | no | no | no | yes | no |
| P4 | yes | yes | no | no | yes | no | yes | yes |
| P5 | yes | yes | no | yes | yes | no | yes | yes |
| P6 | yes | yes | no | no | yes | yes | no | yes |
| P7 | yes | yes | no | no | yes | yes | no | yes |
| P8 | yes | yes | yes | yes | yes | yes | no | yes |
| P9 | yes | yes | no | no | yes | yes | no | yes |

Table 2: Visibility and accessibility description of the tiles in the different solutions.

## References


AbleGamers. (2020). *The ablegamers charity: Our stories*. AbleGamers. Retrieved January 15, 2021, from https://ablegamers.org/pages/our-stories/

Allman, T., K. Dhillon, R., A. E. Landau, M., & Hastuti Kurniawan, S. (2009). Rock vibe: Rock band® computer games for people with no or limited vision, 51–58. https://doi.org/10.1145/1639642.1639653

Andrade, R., Rogerson, M. J., Waycott, J., Baker, S., & Vetere, F. (2019). Playing blind: Revealing the world of gamers with visual impairment. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 116:1–116:14. https://doi.org/10.1145/3290605.3300346

Atkinson, M. T., Gucukoglu, S., Machin, C. H. C., & Lawrence, A. E. (2006). Making the mainstream accessible: Redefining the game, 21–28. https://doi.org/10.1145/1183316.1183321

Calleja, G. (2010). Digital games and escapism. *Games and Culture*, *5*(4), 335–353. https://doi.org/10.1177/1555412009360412

DOWINO. (2016). *A Blind Legend*. Retrieved December 18, 2020, from http://www.ablindlegend.com

EA Tiburon. (2020). *Madden nfl 20*. Retrieved December 19, 2020, from https://www.ea.com/games/madden-nfl/madden-nfl-20

Game Accessibility Guidelines. (2019). *Ensure screenreader support, including menus & installers: Game accessibility guidelines*. Retrieved June 22, 2019, from http://game%5C-accessibility%5C-guidelines.com/ensure-screenreader-support-including-menus-installers/

Harmonix. (2007a). *Guitar hero*.

Harmonix. (2007b). *Rock band*.

Naughty Dog. (2020). *The Last of Us Part II*. Retrieved December 18, 2020, from https://www.playstation.com/en-us/games/the-last-of-us-part-ii/

NetherRealm Studios. (2019). *Mortal kombat 11*. Retrieved December 19, 2020, from https://www.mortalkombat.com

Norman, D. A. (2002). *The design of everyday things: Revised & expanded edition*. Basic Books, Inc.

Porter, J. R. (2014). Understanding and addressing real-world accessibility issues in mainstream video games. *SIGACCESS Access. Comput.*, (108), 42–45. https://doi.org/10.1145/2591357.2591364

RARECSM. (2019). *Sea of thieves: Configuring 'let games read to me' game transcription*. Microsoft Corporation. Retrieved October 2, 2020, from https://support.seaofthieves.com/hc/en-gb/articles/360022122074--Configuring-Let-Games-Read-to-Me-Game-Transcription

Skizzix. (2009). *Shredder Chess*. Retrieved December 18, 2020, from http://skizzix.com/games-for-iphone/shredder-chess/

Strasser, K. (2018). *Subwords*. Retrieved December 18, 2020, from http://subwords.app/

Strasser, K. (2021). *Unity accessibility toolkit: Enhancing accessibility of video games for people with vision impairments*. Master's Thesis.

Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge University Press. https://doi.org/10.1017/CBO9780511816338

Thatcher, J. (1994). Screen reader/2: Access to os/2 and the graphical user interface. *Proceedings of the First Annual ACM Conference on Assistive Technologies*, 39–46. https://doi.org/10.1145/191028.191039

Vice News. (2019). *This is how to play video games if you're totally blind*. Youtube. Retrieved June 13, 2019, from https://www.youtube.com/watch?v=aX0oPwQPo9A

Yuan, B., & folmer eelke, e. (2008). Blind hero: Enabling guitar hero for the visually impaired. *ASSETS'08: The 10th International ACM SIGACCESS Conference on Computers and Accessibility*, 169–176. https://doi.org/10.1145/1414471.1414503