Artificial Intelligence in Video Game Development with Accessibilities in Godot Engine

Jonatan L. Guzmán Flores [1], Segundo E. Cieza-Mostacero [1]

[1] *Cesar Vallejo University, Avenue Larco 1770, Trujillo, Peru*

Abstract **- The objective of the research is to seek the development of accessibility in video games developed in Godot Engine, through the creation of a plugin that allows implementing customizable filters and options instantly, a pure experimental design is used, the software was developed using the SUM methodology. 30 projects from beginning developers were evaluated, collecting data on time, costs and satisfaction through control sheets and surveys, Mann Whitney U tests were also applied due to the data obtained. The results showed a 79% reduction in implementation and development times, a total elimination of extra development costs and a 60% improvement in the identification of elements, functions and environment after using the plugin. This demonstrates that using tools allows for the creation of simple and efficient accessibility options, expanding the options for developers to create more accessibility games.**

Keywords **- Color blindness, tool, Artificial Intelligence, Godot Engine, SUM metodology.**

DOI: 10.18421/TEM144-47
https://doi.org/10.18421/TEM144-47

Corresponding author: Jonatan L. Guzmán Flores,
Cesar Vallejo University, Avenue Larco 1770, Trujillo, Peru
Email: jguzmanfl@ucvvirtual.edu.pe

Received: 05 February 2025.
Revised: 09 September 2025.
Accepted: 03 October 2025.
Published: 27 November 2025.

1. Introduction

Color blindness is a condition that prevents people from seeing certain colors normally. This condition has different variants, with three being the most common: deuteranomaly (difficulty perceiving green tones), protanomaly (difficulty perceiving red tones), and tritanomaly (difficulty perceiving blue tones) [1].

According to Color Blind Awareness, approximately 1 in 12 men (8%) and 1 in 200 women worldwide have some form of color blindness, which translates to around 300 million people globally [2], although this condition is not considered a visual disability, it can limit certain leisure activities, such as watching movies or playing video games [3].

Video games have grown significantly among the general public. According to PwC, in 2023, the gaming industry's revenue reached 262 billion dollars, with projections to increase to 312 billion dollars by 2027 [4]. This increase is mainly influenced by the rise of smaller studios or indie developers, the development of double-AA projects, and the improvement in production quality.

However, accessibility for people with visual impairments, such as color blindness, remains a challenge due to the need for early-stage planning in a project. Additionally, for some projects, it is unfeasible to create options that do not affect the message, vision, or direction of the project. As a result, in indie, double-A, and triple-A developments, accessibility is often an option that ends up being discarded [5].

The gaming industry has always included accessibility options, as they significantly improve the experience for many players, with or without visual impairments. Examples include Fortnite, Call of Duty, and Warframe, which offer options to customize the visual aspect for players with color blindness [6], among them, Warframe, developed by Digital Extremes, stands out for offering total control over its visual settings since its inception [7].

These measures not only expand the potential market for these video games but also make the experience much more comfortable and adjustable for players with visual impairments [8].

Peru is not falling behind in video game consumption and development. With a growing market, more developers are looking to enter the industry and compete internationally. Becket Barrionuevo, Sales Manager at Acer Peru, mentions that according to the Peruvian Association of Electronic Sports and Video Games, the video game sector in Peru generated 120 million dollars in sales in 2021 [9].

However, video games often focus heavily on presenting novel and entertaining ideas to attract attention but tend to overlook customization and player comfort. Despite bringing interesting proposals to the table, they are not entirely customizable for all players, especially when it comes to being inclusive with accessibility options [10].

Godot Engine has stood out as a fast and easy-to-learn game development tool [11] since its release in 2014, Godot has built a large community of developers who appreciate its flexibility and business model, its great potential for developing complex systems like other IDEs [12] and the fact that it is open-source and free software. This allows developers to create 2D and 3D games while focusing on simplicity and rapid development [13].

Despite its many advantages, implementing accessibility options in Godot requires prior project analysis or experience in developing customization options, which increases development time and costs.

Therefore, the implementation of an innovative solution using Artificial Intelligence to simplify the development of accessibility features in projects made with Godot Engine became clear.

This solution would be a tool installed within the IDE, accessible from the assetlib. By providing efficient and quick assistance in creating simple accessibility options for players, the goal is to increase the number of video games developed in Godot with accessibility features.

The main objective of the research is to facilitate the development of accessibility features for people with color blindness in the Godot Engine. The specific objectives of the research cover some crucial aspects, such as:

(a) Increasing the ability to identify key elements that colorblind people face when playing video games,
(b) Reducing the time to develop accessibility options for colorblind individuals, and
(c) Lowering the development cost of accessibility options for developers.

Finally, the respective hypotheses of the research are mentioned. Regarding the development time indicator, the hypothesis:

H1: An Artificial Intelligence reduced the development time of accessibility tools in Godot Engine games.

Regarding the development cost indicator, the hypothesis:

H2: An Artificial Intelligence lowered the development costs of accessibility features in games developed in Godot.

Regarding the element recognition level indicator, the hypothesis:

H3: The AI-based tool improves the recognition of elements in video games developed in Godot.

2. Related Works

This section includes some previous research based on the use of artificial intelligence tools for both software development and as a proposed solution for certain aspects of color blindness, from which information is gathered for conducting the research. It also defines indicators for the tests performed and later compares the results.

The research by [14] titled “Daltonizer: A CNN-based Framework for Monochromatic and Dichromatic Color-Blindness” aims to develop a framework based on convolutional neural networks (CNN) to help people with color blindness distinguish certain colors much more effectively, focusing on correcting and enhancing color perception.

The authors directed their research toward colorblind individuals, and the results showed that the proposed framework successfully corrected and improved color perception, allowing individuals to distinguish colors that were previously difficult to see.

CNN-based architecture distinguishes between different classes of images with an accuracy rate of 82%, which could inevitably reflect the global environment. This demonstrates that the use of image processing techniques and CNNs is effective in improving the visual experience for colorblind individuals.

The research highlights the importance of advanced technologies in addressing color perception issues, emphasizing that innovative solutions can significantly improve the quality of life for people with color blindness. The contribution of the research lies in offering the indicator "Clear identification of elements," as well as various methodologies and frameworks for the development of theartificial intelligence tool.

The research by [15] titled "Measuring GitHub Copilot's Impact on Productivity," analyzes the impact of GitHub Copilot on developer productivity. This tool helps autocomplete code based on the context in which it is being developed.

The research reviews the acceptance rate among developers and how the level of experience influences this. The results, obtained through surveys conducted with over 2,000 developers, show an improvement of between 15.3% and 26.5% in development, as it helps eliminate repetitive tasks and speeds up the completion of known tasks or issues. The contribution to the research was the development time indicator.

The research by [16] presents the development of an AI-assisted technology for people with visual impairments. The main objective is to recognize different objects and provide real-time auditory information, which enhances the individual's understanding of their environment. The study population includes people with visual impairments, and a deep learning model has been trained with multiple images of objects relevant to them.

The results show an average accuracy of 95.19% and 99.69% in object detection and recognition, respectively. Furthermore, the conclusion highlights the effectiveness of the proposed system compared to previous methods that focused only on obstacle detection and location tracking without using deep learning. The contribution of the research is the methodology for determining result quality, as well as the indicator "Clear identification of elements."

The research by [17] titled "Towards cost-effective and robust AI-microservice deployment in edge computing environments," aims to reduce implementation costs in AI-based microservices.

To achieve this, they proposed a scheme involving multiple neural network models such as TDDP (Three-Dimension-Dynamic-Programming) and the development of instance placement strategies to minimize the impact of failures in edge nodes (TLLB, Three-Level Load Balance).

Their results showed that, compared to approaches like DFS (Depth-First Search), TDDP reduced implementation costs by an average of 67% with simple methods, 35-41% with more complex approaches, and 73% in specific scenarios. The contribution to the research was identifying the development cost indicator.

3. SUM Methodology

For the software development, the SUM methodology was chosen. Although it is designed for game development, this does not diminish its agile nature and easy-to-follow steps, providing constant progress tracking and allowing adjustments based on feedback during development [18] making this methodology an excellent choice for this occasion.

3.1. Concept

The first phase is to define the software concept, where the objective is set, which in this case is to facilitate the development of accessibility features in video games. The target audience is also defined, which in this case is developers with intermediate experience in Godot. Finally, the key software requirements must also be identified (Table 1).

Table 1. Functional requirements

	Request	Description
FR01	Color Blind Filter	The developer can create a Deuteranopia, Protanopia and Tritanopia filter, which can be applied as an accessibility option for players or used for color tests during development
FR02	Color Replacement	The developer must also have the ability to create an option to modify any previously selected color in the game scenes, offering a flexible system to adjust the colors based on the preferences or accessibility needs of the players.
FR03	Highlight Objects	The developer should be able to create an option to add a previously selected highlight effect of any scene in the video game
FR04	Node Reading	For the tool to function effectively, it must be capable of recognizing the properties of the scenes it modifies and implementing the accessibility options without interfering with existing configurations or settings.
FR05	Menu Creation	Each option that the tool allows us to create must generate a simple interface with the options selected by the developer, as well as the name, and must be easily located within the project.

3.2. Planning

In this step, software development is organized, maintaining a clear order. For the development of this software, the work was divided into 3 modules based on the 3 key functionalities that the plugin had to have: the colorblind filter, the color replacement system, and the color highlighting system.

This decision was made because the filter is the easiest tool to create due to previous studies that provide assistance for development, as well as being the main tool to test the efficiency of the other functionalities.

Regarding the color replacement and highlighting systems, they have similar structures, as the algorithm is responsible for traversing the project scene nodes to implement the functions without interrupting any previous configurations made by the developer. Therefore, the chosen order was to first develop the color replacement system and then the highlighting system.

3.3. Elaboration

Once the order and parameters for the software development are defined, the development process begins. During this stage, the main functions of the software are completed, aiming to have an early version of the tool in order to conduct tests and identify bugs while continuing with the development.

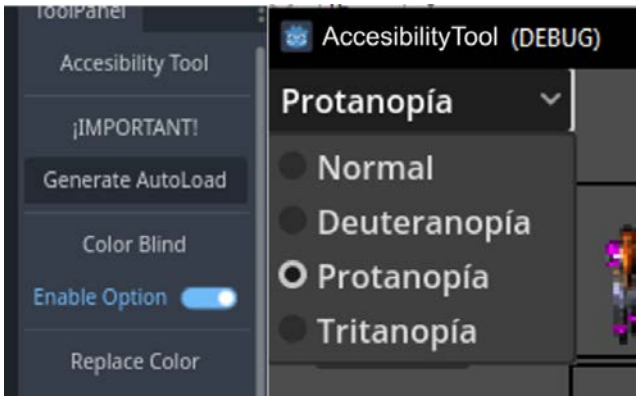

Figure 1. Color blindness filter function

Figure 1 shows the button to generate the autoload and the option to activate the colorblindness filter function. The autoload is important because, in Godot Engine, it allows changes or configurations to be maintained within the game. If it is not generated, the plugin creates the autoload when the options are created.

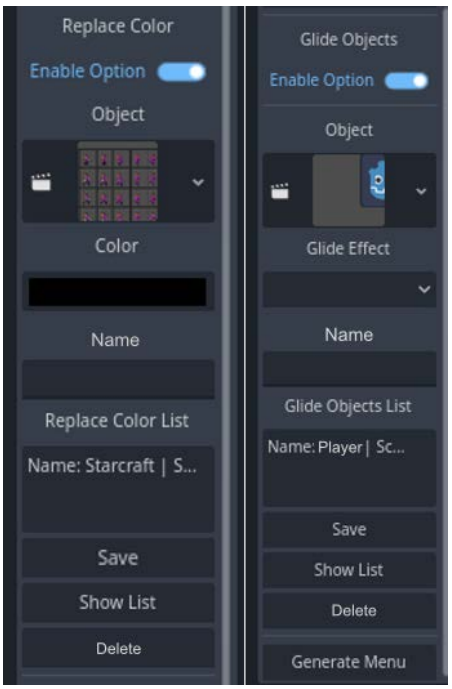

Figure 2. Replace color and highlight object function

Figure 2 shows the options to replace color and highlight objects. In Godot, video game development is managed through scenes, which can represent environments, enemies, characters, etc. These two options allow the developer to select a scene for modification. In the case of "Replace Color," the developer can choose a color for the selected scene. When the function is generated, a menu will be created that allows interaction with the selected scenes. In this example, a sprite is shown with a pink color. (Figure 3).

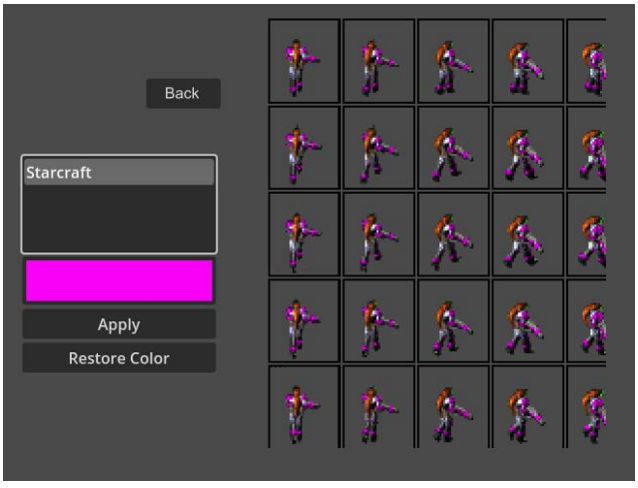

Figure 3. Replace color menu (sprite with original color)

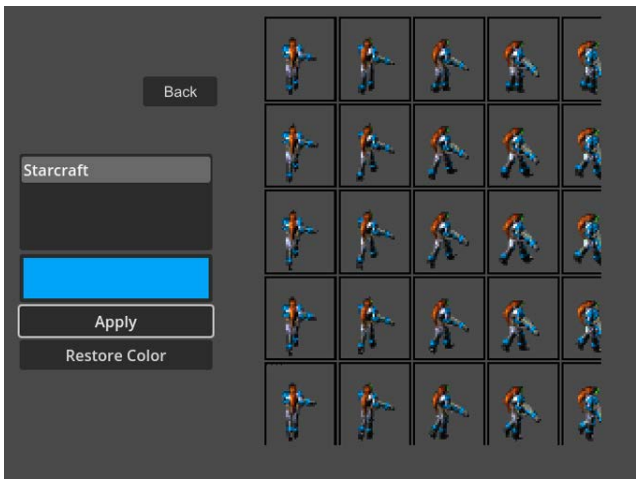

Figure 4. Replace color menu (changed color)

Using the tool, it is possible to create a function that detects this color and allow the player to change it to any desired color (Figure 4).

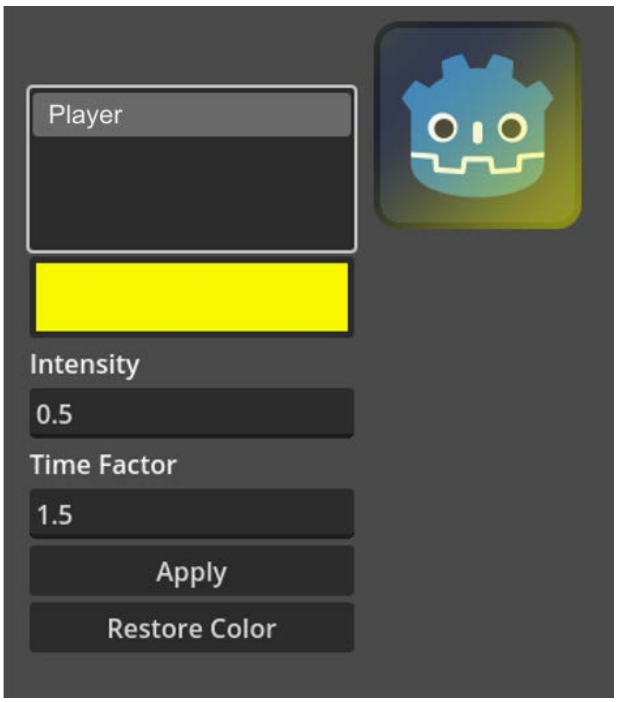

Figure 5. Menu highlight object

In the case of the "Highlight Object" function, it is an option that creates a feature allowing the player to add a lighting effect and modify both the color and certain properties of the effect. In the menu generated for the player, it allows them to activate and deactivate the effect, as well as modify the color, intensity, and speed of the animation (Figure 5).

3.4. Beta

There were only 2 critical functionality errors. The first error affected screens with HD resolutions or 1280 x 720, and it prevented the plugin menu from scrolling when reaching a certain configuration limit.

To fix this, the parent node used to develop the plugin had to be changed to one that allowed scrolling, which caused several interface errors that were corrected later that same day.

The second error occurred because the color replacement function created a canvas layer above the objects with that function and changed the color in real time without affecting other objects. This function started consuming a lot of resources when the selected object was instantiated multiple times at the same time, as the canvas layer was also replicated, which created double the entities with each instance. To fix this, the color change system was switched to one that uses shaders from the parent node and affects the nodes with visual materials within the scene.

3.5. Closing

Once all the errors were corrected, the first final version was released, which was used for testing and data collection in the research. An important point to note is that many of the errors, both major and minor, were related to node compatibility and their reading. Therefore, it is crucial to consider this for future improvements.

4. Methodology

The target population for the research consists of developers who use Godot Engine for game development, including developers with little experience in using Godot. The projects consist of both demos and technical tests. The sample size will be 60 developers, divided into 30 for the control group and 30 for the experimental group.

For data collection, a stopwatch, surveys, and forms were used. The stopwatches measured the time required to implement accessibility features before and after using the tool, the surveys assessed the developers' perception of the effectiveness and ease of use of the solution, and the forms were used to gather data on the impact on the development budget.

Data analysis was conducted using descriptive and inferential statistical techniques [19]. Descriptive techniques included measures of central tendency and dispersion to describe the characteristics of the sample and the results obtained.

To determine the significance of the differences observed before and after the use of AI applications, the Shapiro-Wilk normality test was applied for smaller than 50 samples, due to the control and experimental groups. The non-parametric Mann-Whitney U test was used to assess the significant difference between the means of both groups for the same variable [20].

For the analysis of the data, the Jamovi software was used, which offers an efficient way to conduct statistical analysis and visualize data.

This tool was employed to perform normality tests, non-parametric tests, and generate general statistical summaries of the data, providing clear insights into the results obtained.

5. Results

After the tests, data was collected regarding the implementation of the plugin. With a population of 30 projects, the data was organized based on the average and the research objective, as well as analyzed using both descriptive and inferential analysis.

5.1. Descriptive Analysis

Table 2 presents the data obtained for the indicator "average development time," measured in hours. For the research, a target time of 3 hours was proposed. Subsequently, a minimum time of 6 hours and a maximum of 16 hours were recorded without using the tool, while a minimum of 1 hour and a maximum of 4 hours were achieved when using the tool.

Table 2. Development time indicator

	Average CG	Average EG	Goal	N Greater than average			% Greater than average		
				EG	Goal	CG	EG	Goal	CG
Average development time	9,1	1,9	3	9	24	30	30	80	100

The control group (CG) achieved an average of 9.1 hours to implement one of the common accessibility features without the help of the plugin, while the experimental group (EG) achieved an average of 1.9 hours with the help of the plugin. Based on the implementation results, it is observed that 30% of the results are below the EG average, and 80% are below the target time.

Table 3 presents the data obtained for the "Development Costs" indicator, measured in dollars. The research set a goal for developers to spend $0 on implementing accessibility features.

Subsequently, the control group had a minimum investment of $1 and a maximum of $5, while the experimental group had no investment at all, resulting in both its minimum and maximum values being $0.

The control group (CG) had an average cost of $1.6 for 2D assets, while the experimental group (EG) had an average cost of $0. Based on the implementation results, it was observed that 100% of the outcomes were below both the EG average and the established goal.

Table 3. Development costs indicator

	Average CG	Average EG	Goal	N Greater than average			% Greater than average		
				EG	Goal	CG	EG	Goal	CG
Development costs	1,6	0	0	30	30	30	100	100	100

Regarding the "Level of Element Recognition" indicator in the implementation of visual accessibility in Godot Engine, recognition was measured across three key aspects of the video game: environmental understanding, element identification, and function comprehension. The level was evaluated using the categories: very poor, poor, fair, clear, and very clear, assigning a score from 1 to 5.

Using the Likert scale, three levels of object recognition were designated: low, with a score from 3 to 6; medium, with a score from 7 to 10; and high, with a score from 11 to 15. The control group obtained an average score of 7.7 points, while the experimental group achieved an average of 12.43 points. 100% of the video games in the control group reached a medium level of object identification, whereas 100% of the video games in the experimental group attained a high level of object recognition (Table 4).

Table 4. Element recognition indicator

	Level	Frequency		Total %		General Average	
		CG	EG	CG	EG	CG	EG
Level of Element Recognition	Low	0	0	0	0	7,77	12,43
	Medium	30	0	100	0		
	High	0	30	0	100		

5.2. Inferential Analysis

This analysis comprises two crucial steps: the normality test and the hypothesis test. The normality test was based on the null hypothesis (H0) and the alternative hypothesis (H1). H0 states that the data follows a normal distribution, while H1 asserts that the data does not follow a normal distribution. Based on these hypotheses, it is established that: If the hypothesis test results in $P < 0.05$, the null hypothesis is rejected in favor of the alternative hypothesis. Conversely, if the test results in $P > 0.05$, H0 is accepted, and H1 is rejected.

Regarding the average development time (TPD), the results show a 79% difference in development and implementation time for accessibility tools, favoring the outcomes after implementing the plugin in Godot Engine. When applying a normality test, it is found that the data does not follow a normal distribution (Table 5).

Table 5. Normality test ADT

Group	Statistical	Value
CG	0.902	0.009
EG	0.834	< .001

By using the Mann-Whitney U test and obtaining a significance value lower than 0.05 (Table 6), it is confirmed the success in reducing development times.

Table 6. Hypothesis test ADT

	Test	Statistical	Value
ADT	Mann-Whitney U	0.00	< .001

Continuing with the development cost (CD) indicator, the results show a 100% difference in accessibility development costs, favoring the outcomes after implementing the plugin in Godot Engine.

When applying a normality test, it is found that the data does not follow a normal distribution (Table 7).

Table 7. Normality test DC

Group	Statistical	Value
CG	0.762	< .001
EG	NaN	NaN

By using the Mann-Whitney U test and obtaining a significance value lower than 0.05 (Table 8), it is confirmed the success in reducing development costs.

Table 8. Hypothesis test DC

	Test	Statistical	Value
DC	Mann-Whitney U	210	< .001

Finally, with the Element Recognition Level (ERL) indicator, the results show a 60% difference in favor of the outcomes after implementing the plugin in Godot Engine. When applying a normality test, it is found that the data does not follow a normal distribution (Table 9).

Table 9. Normality test LER

Group	Statistical	Value
CG	0.774	< .001
EG	0.828	< .001

By using the Mann-Whitney U test and obtaining a significance value lower than 0.05 (Table 10), it is confirmed the success in improving the level of object identification.

Table 10. Hypothesis test LER

	Test	Statistical	Value
LER	Mann-Whitney U	0.00	< .001

6. Discussion

Regarding the first indicator, "Average Development Time," in the implementation of visual accessibility in Godot Engine, an average of 9.1 hours was recorded to implement one of the common accessibility features without the help of the plugin, and an average of 1.9 hours with the plugin's assistance. This resulted in a 79% reduction in the average development time for these features, all thanks to the plugin automating many of the factors that cause conflicts or errors when working with color changes and recognition through shaders on various types of objects.

The research by [15] analyzes the impact of GitHub Copilot on developer productivity, a tool that helps autocomplete code in the context of the development environment.

The study examines the acceptance rate among developers and how their level of experience influences this. The results obtained from surveys conducted with over 2,000 developers show an improvement in development productivity ranging from 15.3% to 26.5%, as it helps eliminate repetitive tasks and accelerates the completion of known tasks or problems.

In comparison to the research, the tool achieved a higher percentage of time reduction, as it not only focuses on code creation but also on modifying 2D assets of the project, even generating add-ons like menus. While itis undeniable that GitHub Copilot is a great tool that works in any type of development, unlike the tool proposed in the research, aiming to reach that level of complexity is a recommendation to consider.

The fact that most of the projects were top-down or platform games werw also highlighted, so the majority of the art and identity were focused on the gameplay rather than the menus, which were usually the elements that needed to be adapted the most after being generated with the tool. Larger projects with thematic menus or those in line with the artistic decisions of the project will likely take some time to modify the auto-generated menus or take the generated code and shaders and adapt them to their already implemented menu system.

Regarding the "Development Costs" indicator in the implementation of visual accessibility in Godot Engine, the control group had an average of $1.6 in 2D asset costs, while the experimental group had an average of $0, showing a 100% reduction in costs. This is because the plugin allows the developer to select the objects that can be managed by the player, eliminating the need to invest in new commissions, as graphical changes are made by modifying the color on screen in real-time.

In the research by [17] a framework is proposed involving multiple neural network models such as TDDP (Three-Dimensional-Dynamic-Programming) and the development of instance placement strategies to minimize the impact of failures in edge nodes TLLB (Three-Level Load Balance).

Their results showed that compared to approaches like DFS (Depth-First Search), TDDP reduced implementation costs by an average of 67% with simple methods, 35-41% with more complex approaches, and 73% in specific scenarios. In comparison to this study, the reduction in costs achieved was somewhat lower, indicating a greater reduction in costs in this case.

It is important to highlight that the projects used for the research were demos and technical tests, so most of them had costs related to art, costs that disappear upon implementing the tools because the need to implement different color models is eliminated by leaving everything to user customization. This type of video game also influences this; in larger projects like visual novels or graphic adventures, the cost difference would be reflected in the implementation time, since these games usually do not focus as much on the menu section, and therefore, the tool automates those sections.

It is also important to mention that, in comparison to the research "Towards cost-effective and robust AI-microservice deployment in edge computing environments," the proposed tool has a specific purpose, while the use of TDDP and TLLB is employed in general software development. As seen in their results, the more specific the application is, the greater the level of cost reduction.

Regarding the "Object Recognition" indicator in the implementation of visual accessibility in Godot Engine, the average score for the control group was 7.7 points, while the experimental group had an average of 12.43 points, showing a 60% improvement in the experimental group compared to the control group. This is because, in similar projects, a color filter is often chosen, whereas it is much better to provide aids beyond just a filter. Modifying colors or ways to highlight objects are the reasons why this plugin significantly helps small projects, as well as allowing the player to choose how to modify them.

In the research by [14] on image modification using convolutional neural networks for better color understanding, a result of 82% improvement in accuracy when pointing out objects was achieved. This is a significant improvement that, in comparison to the research findings, is not far off, especially considering that neural networks are a much more complex and powerful technique.

In the research by [16] on improving object recognition, the results show an average accuracy of 95.19% and 99.69% in object detection and recognition, respectively, with low temporal complexity that allows for real-time environmental perception. These results demonstrate that simply changing colors is not enough for consistent improvement, but additional aids such as identification points, shapes, or customizable markers are necessary.

It is also important to clarify that this level of improvement depends on the complexity or nature of the project. In "Bullet Hell" games, which have a high level of projectiles, more types of tools are required for significant improvement, such as UI customization. Meanwhile, in projects like visual novels or graphic adventures, the majority of elements are character portraits and text bubbles, so the improvement focuses more on highlighting the interactive aspects for the player.

7. Conclusion

The development of accessibility features in video games developed in Godot Engine was successfully facilitated by the tool. The algorithm effectively implements solid, uncomplicated options that are ready for use and can be edited and adapted by the developer to suit specific aspects of their game.

A reduction in development time for implementing accessibility options for people with color blindness in video games developed in Godot Engine was achieved. The obtained data shows 9.1 estimated hours for development and implementation of accessibility features before the plugin was implemented, and 1.9 estimated hours for development and implementation after the plugin was implemented.

A reduction in development costs for implementing accessibility options for people with color blindness in video games developed in Godot was also achieved. The obtained data shows an estimated cost of $1.60 for developing accessibility options before the plugin was implemented, and $0 in costs for developing accessibility options after the plugin was implemented.

Lastly, an increase in the level of object identification in the video games/demos used for the plugin tests in video games developed in Godot Engine was achieved. The obtained data shows an average object recognition score of 7.7 points before the plugin implementation, indicating a medium/normal level of object identification, and a 12.4-point average score after the plugin implementation, showing a high level of object identification elements.

Ultimately, the development of this Godot plugin not only validates the initial hypothesis that a specialized tool simplifies the creation of effective accessibility options, but also highlights its practical advantages, it significantly reduces the technical barrier to entry, streamlines the implementation process, and proactively encourages inclusion from the initial design phases.

However, it is important to renognize its limitations; the main disadvantage lies in its current scope, which focuses on specific visual solution, leaving aside other crucial disabilities, and its effectiveness ultimately depends on developer integration, as well as the type of video game. Looking ahead, this work lays the groundwork for the study of many important features to be optimized, incluiding the use of image generation for symbol-based markup, real-time filter modification by players, and the ability to modify sprites or resource to add user-modifiable features.

References:

[1]. Jonauskaite, D., et al. (2021). Colour-emotion associations in individuals with red-green colour blindness. *PeerJ*, *9*, e11180. Doi: 10.7717/peerj.11180.

[2]. Colour Blind Awareness. (n.d.). *About Colour Blindness.* Colour Blind Awareness. Retrieved from: https://www.colourblindawareness.org/colour-blindness/ [accessed: 10 January 2025]

[3]. Frane, A. (2015). A call for considering color vision deficiency when creating graphics for psychology reports. *The Journal of General Psychology*, *142*(3), 194-211. Doi: 10.1080/00221309.2015.1063475

[4]. *PwC España. (2023). Principales conclusiones del Global Entertainment & Media Outlook 2023-2027.* Pwc.es. Retrieved from: https://www.pwc.es/es/entretenimiento-medios/entertainment-media-outlook-espana-2023-2027.html [accessed: 12 January 2025]

[5]. Zhang, X., & Dahu, W. (2019). Application of artificial intelligence algorithms in image processing. *Journal of Visual Communication and Image Representation*, *61*, 42-49. Doi: 10.1016/j.jvcir.2019.03.004

[6]. Ruppert, L. (2018). *Fortnite update adds colorblind options, here's why that's important.* ComicBook.com. Retrieved from: https://comicbook.com/gaming/news/fortnite-update-adds-colorblind-options/ [accessed: 13 January 2025]

[7]. Digital Extremes. *(2023). Abyss of Dagath - Dev Workshop: Accessibility + HUD Improvements.* Warframe Forums. Retrieved from: https://forums.warframe.com/topic/1366783-abyss-of-dagath-dev-workshop-accessibility-hud-improvements/ [accessed: 15 January 2025]

[8]. Ananth, C., et al. (2022). Artificial Intelligence based Visual Aid with Live Tracking of Visually Impaired People. *2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, 573-577. Doi: 10.1109/ICTACS56270.2022.9988614

[9]. López Sevilla, E. (2022). *La expansión de los videojuegos en el Perú*. La Cámara. Retrieved from: https://lacamara.pe/la-expansion-de-los-videojuegos-en-el-peru/ [accessed: 02 February 2025]

[10]. Tillem, M. (2023). *Color blindness in the digital gaming landscape: Addressing critical issues and research gaps. Proceedings of the 17th European Conference on Games Based Learning (ECGBL 2023)*. Doi: 10.34190/ecgbl.17.1.1749

[11]. Van Rozen, R. (2023). Game Engine Wizardry for Programming Mischief. *Proceedings of the 2nd ACM SIGPLAN International Workshop on Programming Abstractions and Interactive Notations, Tools, and Environments*, 36-43. Doi: 10.1145/3623504.3623570

[12]. De Pontes, R. G., Gomes, H. M., & Seabra, I. S. R. (2022). Particle swarm optimization for procedural content generation in an endless platform game. *Entertainment Computing*, *43*, 100496. Doi: 10.1016/j.entcom.2022.100496

[13]. MMMAcademy. (2025). *Motor Godot: Guía completa para crear videojuegos 2D y 3D con una herramienta gratuita y de código abierto.* MMMAcademy. Retrieved from: https://mmmacademy.es/godot-que-es-ventajas-funcionamiento/ [accessed: 04 June 2025]

[14]. Rathee, D., & Mann, S. (2022). Daltonizer: A CNN-based Framework for Monochromatic and Dichromatic Color-Blindness. *2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST)*, 1-5. Doi: 10.1109/AIST55798.2022.10065004

[15]. Ziegler, A., et al. (2024). Measuring github copilot's impact on productivity. *Communications of the ACM*, *67*(3), 54-63. Doi: 10.1145/3633453

[16]. Joshi, R. C., et al. (2020). Efficient multi-object detection and smart navigation using artificial intelligence for visually impaired people. *Entropy*, *22*(9), 941. Doi: 10.3390/e22090941

[17]. Wu, C., et al. (2023). Towards cost-effective and robust AI microservice deployment in edge computing environments. *Future Generation Computer Systems*, *141*, 129-142. Doi: 10.1016/j.future.2022.10.015

[18]. Acerenza, N., et al. (2009). *Una metodología para desarrollo de videojuegos: versión extendida*. PEDECIBA.

[19]. Hartas, D. (2010). *Educational Research and Inquiry: Qualitative and Quantitative Approaches*. Bloomsbury Academic. Doi: 10.5040/9781474243834

[20]. Sheskin, D. J. (2011). *Handbook of parametric and nonparametric statistical procedures* (5th ed.). Chapman and Hall/CRC. Doi: 10.1201/9780429186196