Exploring Visual and Multimodal Interaction in NCL Authoring

Paulo Victor Borges
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
pvborges@telemidia.puc-rio.br

Daniel de S. Moraes
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
danielmoraes@telemidia.puc-rio.br

Joel dos Santos
CEFET-RJ
Rio de Janeiro, Brazil
jsantos@eic.cefet-rj.br

Débora C Muchaluat-Saade
MídiaCom Lab
Fluminense Federal University
Nitéroi, Brazil
debora@midiacom.uff.br

Sérgio Colcher
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
colcher@inf.puc-rio.br

ABSTRACT

This paper introduces two innovative tools for enhancing interactive multimedia authoring using the Nested Context Language (NCL): (i) a visual extension that supports more traditional interactions with mouse and keyboard and (ii) a multimodal extension that incorporates gesture recognition and voice commands. These tools were implemented as Visual Studio Code extensions and aim to streamline the editing process, making it more intuitive and accessible. We present an evaluation of the usability and acceptance of both tools with developers in an experiment with three tasks for creating and manipulating spatial regions in hypermedia documents. By exploring the potential of multimodal interfaces, this work sets the stage for more efficient and user-friendly document editing.

KEYWORDS

Authoring, LLMs, NCL, Code Generation, Visual Studio Code

1 INTRODUCTION

Digital TV (DTV) applications [26] have revolutionized content delivery and consumption, offering a rich multimedia experience that integrates text, images, audio, and video. These applications extend beyond passive viewing by enabling interactivity and creating a more engaging and immersive environment for the viewer.

DTV applications (as well as other interactive multimedia applications) can be developed using structured content, known as *multimedia/hypermedia documents*. The structure of these documents has been the goal of standardization efforts in many instances, including the definition of declarative languages like the NCL (Nested Context Language) [28], a language based on the NCM (*Nested Context Model*), which facilitates the logical structuring of hypermedia documents through compositions and specifies spatio-temporal relationships using connectors and links within an event-based paradigm.

Current digital TV scenarios require simple and quick methodologies to create interactive multimedia applications. As in other scenarios, there is a growing need for functionalities that can assist programmers in developing these applications efficiently. Thus, authors with varying skills might be able to create correct and functional applications.

Therefore, creating and editing multimedia content, including DTV applications, require precise and efficient tools that can effectively manage various elements within a digital interface while also maintaining ease of use. Traditional tools that have been relying on mouse and keyboard interactions have proven their effectiveness, but there is still considerable room for improvement. Natural user interfaces (NUIs) [14, 17], such as body movements and gestures, have been used in many settings to operate machines, communicate with intelligent environments, and control smart home appliances [5] and could also be applied in a multimedia context to create more intuitive and engaging authoring experiences.

Large Language Models (LLMs), on their turn, have recently revolutionized various fields, enabling their use in developing a wide range of applications, such as chatbots (like chatGPT and Gemini) that can handle queries in different contexts. These models have also shown good results in tasks related to programming and code synthesis [7, 8, 19, 23].

In this sense, this work proposes two distinct tools to help improve the development of interactive applications for Digital TV, specifically in creating the layouts defined by the region base of a document. The first is a visual tool that allows the developer to visualize the regions defined in a document's region base and relies on traditional mouse and keyboard interactions to create and manipulate regions graphically. The second tool incorporates gesture recognition and voice commands, processed by a language model to generate editing actions (in addition to the visualization of the created regions). These tools are implemented as Visual Studio Code (VS Code) extensions [9], taking advantage of its robust feature set, versatility, and adaptability.

We also present an evaluation of the usability and acceptance of both tools with developers in an experiment involving tasks of creation and manipulation of regions. This evaluation was conducted through a custom usability questionnaire that was inspired by and adapted from the principles of the System Usability Scale (SUS) [4, 6, 18] to fit our specific case.

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2024). Juiz de Fora, Brazil. Porto Alegre: Brazilian Computer Society, 2024.

ISSN 2966-2753

The remainder of this paper is structured as follows: Section 2 provides a review of related work. Section 3 describes the methodology and implementation of the tools. Section 4 details the experimental setup, including the tasks and participants. Section 5 presents the results of the experiment and the usability questionnaire. Finally, Section 6 discusses the findings, implications, and potential future work.

2 RELATED WORK

The development of interactive multimedia applications for digital TV has been significantly advanced by authoring tools that facilitate both textual and visual authoring approaches.

For example, NEXT (NCL Editor Supporting XTemplate) [24] is a graphical editor developed to create NCL documents using hypermedia composite templates specified in the XTemplate 3.0 language [11], making the creation of interactive digital TV applications more accessible to authors without in-depth NCL knowledge. NEXT offers functionalities such as creating and editing NCL documents in different views, utilizing an extensible template library that can be adapted to different skill levels of authors. By providing a graphical interface and a set of plugins, NEXT simplifies the development process and allows authors to focus on the content rather than the intricacies of NCL programming. This tool represents a significant step forward in enabling a broader range of users to contribute to the development of interactive multimedia applications, enhancing both the usability and efficiency of the authoring process.

STEVE (Spatio-Temporal View Editor) [10] allows users without prior knowledge of authoring languages to create interactive multimedia applications for web and digital TV systems, presenting a graphical interface that visually exposes the elements and their relationships and exporting to HTML5 and NCL. These tools often feature user-friendly interfaces, enabling a broader audience to engage in multimedia content creation without the need for extensive technical expertise. This inspires the development of a tool that can generate code through a visual interface while seamlessly integrating into popular development environments.

A notable contribution is the Lua2NCL framework [22], which addresses the verbosity and complexity associated with NCL by providing a set of textual features that reduce the effort required for creating NCL applications. Lua2NCL leverages the Lua scripting language to enable the creation of applications with reduced code size and enhanced readability, making it accessible even to developers with limited experience in NCL. This framework has demonstrated considerable effectiveness in reducing the time and code length necessary for authoring digital TV applications, as evidenced by experiments showing significant reductions in both metrics compared to traditional NCL coding.

NCL Eclipse [25] is a textual integrated development environment designed to assist in developing interactive multimedia applications in NCL. As a plugin for the Eclipse IDE, NCL Eclipse offers user-centred features such as automatic and contextual code suggestion, validation of NCL documents, and syntax highlighting for XML elements and reserved words. By providing functionalities like automatic code formatting and error indication during the authoring process, NCL Eclipse significantly enhances developers' productivity.

The development of authoring tools for interactive Digital TV (iDTV) has focused on improving usability for content creators who may not be programmers. NCL presents a particular challenge for non-technical users due to its complexity. Recent studies, such as those of Moraes et al. [21], highlight the potential of using LLMs to facilitate NCL code generation. However, their initial findings indicate that current pre-trained LLMs struggle to generate high-quality NCL code due to syntax and language rule challenges. This underscores the need for fine-tuning LLMs to more effectively process domain-specific languages like NCL, which could significantly enhance the usability of multimedia authoring tools by allowing users to define application requirements in natural language.

In that context, numerous authoring tools have introduced significant innovations to facilitate the creation of NCL documents through graphical interfaces, reduced code verbosity, and enhanced textual authoring features such as automatic code suggestion and document validation. Inspired by these advancements, our work proposes a visual approach to create and edit applications' layouts defined by the region base, allowing graphical manipulation of those elements using a mouse and keyboard. We also propose a version with interaction enriched by gesture recognition and voice commands, integrated with an LLM tool to generate the region base NCL code. With these tools we aim for an intuitive and efficient approach that can facilitate the creation and edition of regions in an NCL application.

3 METHODOLOGY

To create a tool available in IDEs and useful to users of different experience levels, we sought to develop extensions that facilitate the editing of NCL documents through intuitive visual interfaces. These extensions were designed for both beginners, who benefit from the simplicity of visual manipulation, and advanced users, who need precise control over the document structure. Although technologies exist to assist in the authoring of NCL code, there is still a need for a more modern solution that integrates with current development environments. Our methodology, therefore, focuses on combining traditional and multimodal interaction techniques to improve the user experience when creating and editing NCL content.

In this context, our methodology involves two main aspects: (i) the development of the two extensions for editing regions in an NCL document and (ii) the implementation of an experiment to evaluate these tools with developers with various experience levels. The overall system architecture is designed to provide real-time feedback and seamless interaction for multimedia content editing.

The main components of the systems are presented in the following sections.

3.1 VS Code Extensions

Two custom extensions were developed for VS Code using TypeScript: a Visual Extension and a Visual Multimodal Extension. The Visual Extension is based on the NCL document being edited in the IDE and opens a WebView interface that visually represents the regions described in the <regionBase> element.

Figure 1 illustrates the graphical elements in Visual Studio Code, where a window opens alongside the code editor, allowing regions

to be visually represented in the interface and ready to receive user interactions.

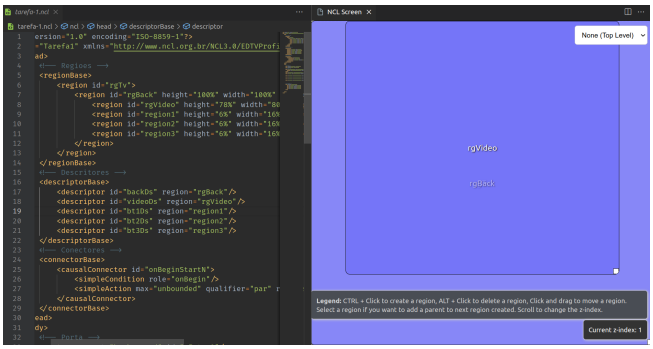

Figure 1: Visual Extension Interface

In the Visual Extension, authors can create new regions, change the z-index, move, resize or delete them using the keyboard and mouse, as illustrated in Figure 2. Any changes made in the graphical interface are reflected in the code, providing an intuitive way to manipulate the NCL document visually.

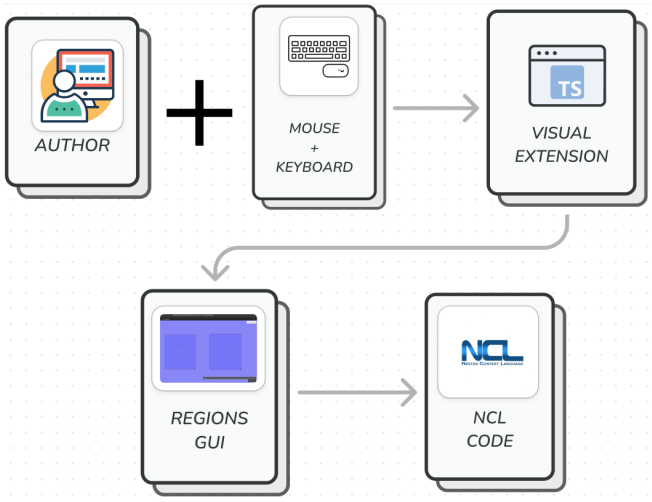

Figure 2: Visual Extension

The Visual Multimodal Extension builds upon the capabilities of the Visual Extension by incorporating gesture recognition and voice commands. As Figure 3 depicts, it communicates with an external multimodal API via WebSocket, which is responsible for capturing gestures and voice commands. These inputs allow users to interact with the regions in the same ways as with the Visual Extension—moving, resizing, deleting, and creating new regions—but with the added convenience of natural user interactions.

Both extensions ensure that the generated code of the region elements adheres to the rules specified in the NCL standards [1], such as parent regions, z-index, and more.

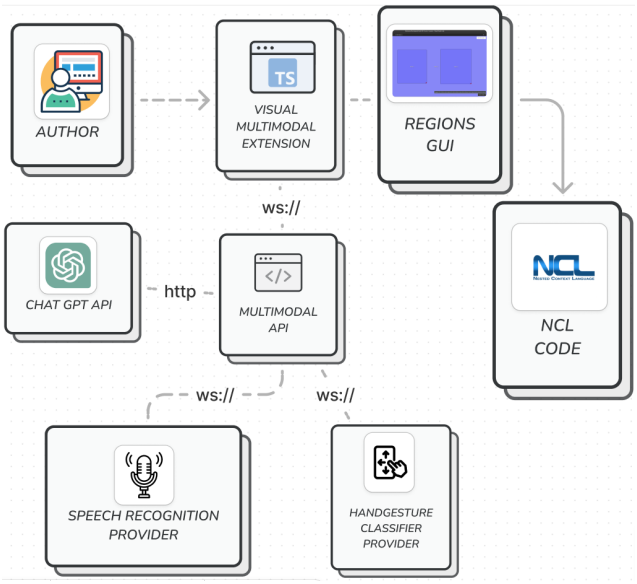

Figure 3: Visual Multimodal Extension

3.2 Multimodal API (MMA)

The Multimodal API (MMA) serves as the central hub for processing and routing data, playing an important role in the seamless integration of multimodal interactions within the NCL authoring tools. It receives messages from the Visual Studio Code extensions and coordinates with external services for gesture classification and speech recognition. Designed with SOLID principles [2, 16, 27], the MMA ensures modularity, maintainability, and scalability, allowing for easy updates and integration of new features without compromising the system's integrity.

To closely align the architecture with business requirements and domain logic, Domain-Driven Design (DDD) practices [3, 20, 20] were applied. This approach facilitated the creation of decoupled components within the MMA, enabling seamless integration with various models and services for gesture and speech recognition, as well as any large language models (LLMs). This design ensures that the system remains highly adaptable to future advancements, allowing the incorporation of new technologies and methodologies as they emerge.

Furthermore, the MMA's architecture supports real-time processing and feedback, crucial for maintaining an interactive and responsive user experience. By adhering to SOLID principles and DDD practices, the MMA meets current technical requirements and also provides a robust foundation for ongoing development and enhancement. This strategic design choice underlines the commitment to creating a scalable, maintainable, and user-centric system that can evolve alongside the rapidly advancing field of multimodal interaction technologies.

3.2.1 HandGesture Classifier Provider. : This component is responsible for capturing video from the user's webcam and applying the classification algorithm provided by the Mediapipe library [13, 31] to identify specific hand gestures made by the user. It detects gestures such as Thumbs Up, open hand, and the hand's position for

movement purposes. The identified gestures are then sent to the Multimodal API (MMA) via WebSocket. The provider sends detailed information about the current gesture, including the type of gesture and the hand's position, ensuring precise and responsive interaction with the NCL authoring tools.

3.2.2 Speech Recognition Provider. : This component is responsible for capturing audio from the user's microphone and applying the Google Speech Recognition API package in Python [30] to convert it to text and identify voice commands. The process begins with the continuous capture of audio input, which is then processed in real-time to transcribe spoken words into text. The Google Speech Recognition API is designed to handle various accents and speech patterns, ensuring high accuracy in the transcription of commands. By breaking down the audio into phonetic units and comparing them against a comprehensive database of known speech patterns, the API generates an accurate textual representation of the spoken commands.

Once the speech is converted to text, the recognized commands are parsed to identify specific keywords and phrases corresponding to predefined actions within the NCL authoring tools. The parsed commands are then formatted into a structured message, including details such as the command type and relevant parameters. This structured message is sent to the Multimodal API (MMA) via WebSocket, enabling real-time processing and interaction within the NCL authoring environment. The use of WebSocket technology ensures low-latency, bidirectional communication, allowing for seamless and responsive updates based on the user's voice commands.

3.2.3 Chat GPT API. : The Multimodal API (MMA) sends HTTP requests to the Chat GPT API for processing recognized gestures and voice commands using the GPT-3.5-turbo model. This language model generates specific editing actions based on the input provided. After each recognition by the speech recognizer, the recognized text is sent to the Chat GPT API along with a carefully crafted prompt to ensure the returned information is accurately processed. This method guarantees that the response is correctly formatted for seamless communication with the VS Code extension, thereby eliminating the need for any additional cleanup of the spoken command.

An example of the prompt used is:

```
You are an intelligent assistant that helps create
and manipulate
regions in a graphical interface.
The user's command is: "{command}"
You are working with a screen resolution of 1920x1080
pixels.
Here are the types of commands you can generate:
1. createRegion - Creates a new region based on the
provided coordinates. Example: Create a region
of 100x100 pixels at position 200,200.

Your job is to interpret the user's command and
generate the appropriate response in correct format.

Please provide a structured response in the following
format:
- Command: createRegion
- left: <value>
- top: <value>
- width: <value>
- height: <value>
- id: <value>
- title?: <value>
- z-index?: <value>

Now, interpret the user's command and provide the
appropriate response, ensuring to use integer values for
coordinates and dimensions.
```

In this context, prompt engineering involves designing and optimizing input prompts to enhance LLM performance and effectiveness. This technique guides LLMs to produce accurate, relevant, and contextually appropriate responses [12]. Carefully crafted prompts allow users to control the model's output, ensuring alignment with specific requirements and objectives.

We used a prompt to enable an LLM to assist in creating and manipulating regions within a graphical user interface (GUI). This structured approach, akin to software patterns, ensures clarity and reusability. The goal is to program interactions between the user and the LLM to efficiently generate NCL code for content editing tasks, with a standard return format that the client application can always interpret correctly. This approach is informed by principles outlined in the prompt pattern catalog by White et al. [29].

This prompt has been adapted for various method calls to achieve different objectives, such as resizing, deleting, and moving regions. Each adapted prompt ensures that recognized voice commands or gestures are accurately interpreted and formatted for seamless communication with the VS Code extension. By tailoring the prompt to handle specific commands, the system efficiently processes a wide range of editing actions, enhancing flexibility and usability.

4 USER EXPERIMENT

A remote experiment was conducted to evaluate the effectiveness and usability of the proposed systems for multimedia application editing. This study explores whether visual and multimodal tools for NCL authoring can enhance the editing experience compared to traditional methods.

After the planning phase a pilot study was conducted, lasting 1 hour via call, to assess the validity of the questions, the quality of the instructions, and the material provided.

Based on the pilot study feedback, corrections and improvements were made, and then an extended experiment was carried out with a larger group of participants. The experiment involved programmers of various experience levels from different universities performing editing tasks using three different interaction methods: (*a*) using only the code editor; (*b*) using the visual extension traditional mouse and keyboard interaction, (*c*) and the visual extension integrated with gesture and voice-based interface.

4.1 Participants

A total of 11 programmers participated in the study. The participants were selected based on their familiarity with multimedia content editing and their experience with programming in VS Code.

4.2 Tasks

The participants were asked to complete a series of basic tasks involving the creation and manipulation of regions within an NCL document. These tasks included:

(1) Creating 5 new regions by specifying dimensions and positions, following the example illustrated in Figure 4.
(2) Resizing the previously created regions to resemble the layout illustrated in Figure 5.
(3) Resizing and moving regions to different positions, as shown in Figure 6.

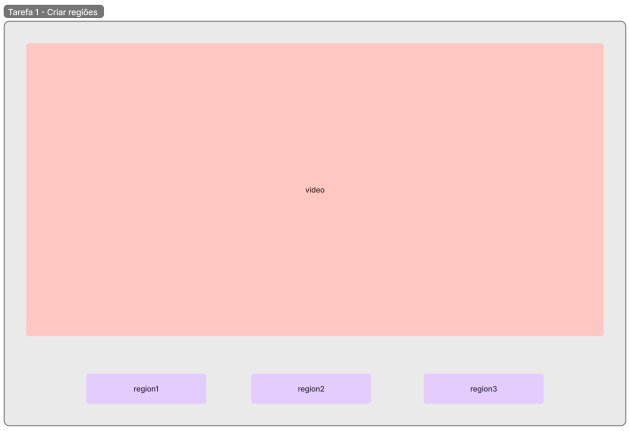

Figure 4: First Task - Create 5 regions that reproduce the defined layout

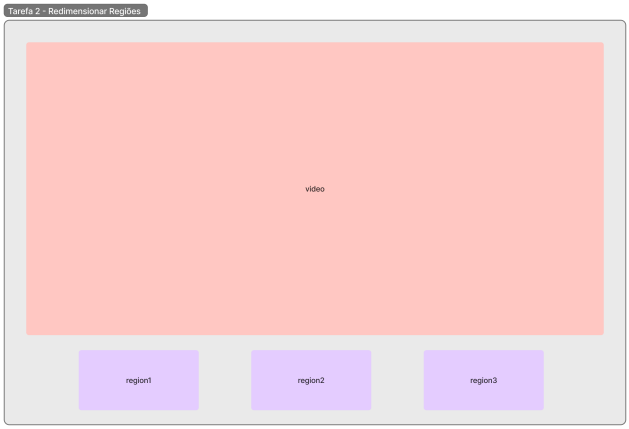

Figure 5: Second Task - Resize existing regions

Each participant performed these tasks in three settings: once using only the VS Code code editor; once using the traditional mouse and keyboard interaction with the visual interface; and once using the gesture and voice-based interaction alongside the visual interface. The experiment always started with the code editor setting, whilst the order of the two other settings was randomized to mitigate any learning effects and ensure that the results were not biased by the sequence in which the tasks were performed.

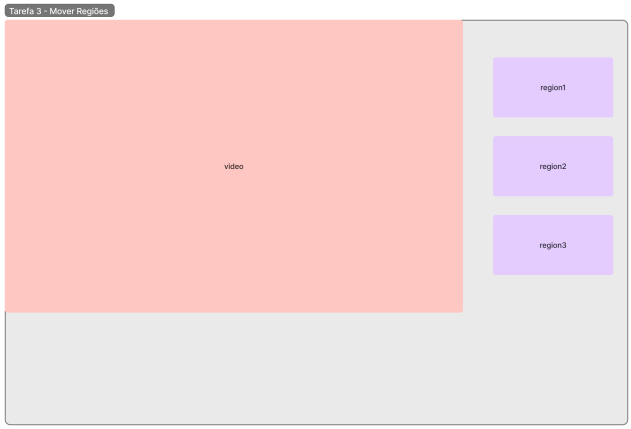

Figure 6: Third Task - Move and resize the regions

4.3 Procedure

The experiment was conducted individually and remotely. Participants were given a script on how to install and use the extensions before starting the tasks. The script also detailed each task and exemplified the expected layouts, as shown in Figures 4, 5, 6. All materials required for the implementation of the tasks were made available, including the incomplete NCL documents, which the participants were meant to fill. After completing each experiment setting, participants were instructed to respond to a questionnaire related to that specific setting.

After completing the tasks of a setting, participants were asked to fill out a questionnaire assessing their experience and the usability of the interactions. The questionnaire included questions on various aspects such as ease of use, efficiency, accuracy, user satisfaction, learning curve, preference, specific feedback, and suggestions for improvement. These questions aimed to gather comprehensive feedback on how easy and intuitive each interface was to use, how quickly tasks could be completed, how accurately tasks were completed, overall user satisfaction, how easy it was to learn to use each interface, preferences between the two interfaces, specific likes or dislikes, and any suggestions for improvement. The questionnaire was inspired by the System Usability Scale (SUS) [4, 18] and incorporates elements from it, although it does not strictly follow the standard SUS format. This adaptation allowed us to focus on particular aspects of usability that were crucial for our study, such as the unique interaction methods of creating and manipulating regions, and to gather more targeted feedback from participants.

Additionally, participants were asked to respond to the following statements about their experience using a Likert Scale [15] from 1 to 7, where 1 represents Strongly Disagree and 7 represents Strongly Agree:

(1) *"I think it is easy to use the tool to create and edit regions."*
(2) *"I was able to create and edit regions using the tool."*
(3) *"I think the regions created and edited with the tool are correct."*
(4) *"I am satisfied with using the tool to create and edit regions."*

These statements helped evaluate the creation and editing capabilities, the accuracy of the regions, and overall satisfaction with the tool.

4.4 Data Collection

The data collection process involved timestamping each participant's responses to track when the data was submitted. Participants' email addresses were the only personal information collected, to ensure the validity of responses and for follow-up purposes. Also, participants' experiences with NCL and VS Code were assessed to understand the correlation between familiarity with these tools and their performance and satisfaction with the interfaces.

Participants preferred interface and their reasons for this preference were noted, providing valuable information on the usability and practicality of the multimodal approach compared to traditional methods. Additionally, participants were asked about their experience with the VS Code IDE and their familiarity with NCL to better understand how these factors influenced their preferences and feedback.

Additionally, participants were invited to submit the final version of their NCL files and indicate if they would like to schedule a meeting for further discussion, allowing for a deeper understanding of their experiences and suggestions for improvement.

5 RESULTS

The collected data were analyzed to compare the performance of the three interaction methods: only using the VSCode text editor, the traditional mouse and keyboard interface (Visual Extension), and the gesture and voice-based interface (Multimodal Extension). Each session lasted approximately 50 minutes, during which participants completed tasks and provided feedback.

Table 1: Summary of Results Comparing Textual, Visual Extension (VE), and Multimodal Extension (MME) Modalities

Metric	Textual (median)	VE (median)	MME (median)
Ease of Use	5.0	7.0	5.0
Accuracy	6.0	7.0	6.0
Overall Satisfaction	5.0	6.0	5.0
Context of Use	General	General	Specific Situations
Preference	0%	0.71%	29%

5.1 Participant Overview

A total of eleven participants took part in the study, with varying levels of experience with NCL and VS Code. However, only seven participants completed the experiment and responded to all survey sections. Therefore, we will use only these seven complete responses for the analysis.

Their experiences ranged from beginners to advanced users, providing diverse perspectives. This diversity in experience levels helped ensure that the feedback covered a broad spectrum of potential users. Participants were primarily students, professionals, and researchers from various fields, ensuring a well-rounded evaluation of the interfaces.

5.2 Performance Metrics

Participants rated the ease of use, efficiency, accuracy, and overall satisfaction for the textual, traditional mouse and keyboard interface (Visual Extension), and the gesture and voice-based interface (Multimodal Extension) on a scale from 1 to 7. In this scale, 1 means "Strongly Disagree" and 7 means "Strongly Agree". Additionally, the preference ratings reflect the number of votes each interface received. The following observations were made, which are detailed below and summarized in Table 1:

- Ease of Use: Participants had the perception that the Visual Extension (VE) interface was generally easier to use, with a median rating of 7.0, compared to 5.0 for both the Multimodal Extension (MME) and the textual interface. This significant difference highlights the steep initial learning curve associated with the multimodal interface. Users thought the VE interface was more intuitive and straightforward, whereas the multimodal interface required more effort.
- Accuracy: Participants reported higher accuracy with the Visual Extension, with a median rating of 7.0, compared to 6.0 for both the Multimodal Extension (MME) and the textual interface. The primary contributors to lower accuracy in the multimodal interface were errors in gesture recognition and voice command interpretation. These inaccuracies often resulted in unintended actions or the need for repeated attempts to achieve the desired outcome, thereby affecting overall task performance.
- Overall Satisfaction: The Visual Extension received higher overall satisfaction scores, averaging 5.9, while the gesture and voice-based interface received a score of 4.3 and the textual interface scored 4.0. Participants appreciated the reliability and ease of use of the traditional interface. However, despite the lower satisfaction score for the multimodal interface, some users expressed excitement about its potential. They highlighted that with further refinements and improvements in gesture and voice recognition accuracy, the multimodal interface could become a powerful and efficient tool for NCL document editing, enhancing user engagement and interaction.

5.3 Qualitative Feedback

Participants provided detailed feedback on their experiences with both interfaces. Key points included:

- Learning Curve: Participants perceived that the gesture and voice-based interface had a steeper learning curve, particularly for those unaccustomed to such interaction methods. Training and practice were required to achieve proficiency. Beginners found the traditional interface more intuitive and less frustrating.
- Context of Use: Some participants suggested that the gesture and voice-based interface could be more effective in specific contexts where hands-free operation is beneficial, such as during live presentations or when physical interaction with a keyboard and mouse is impractical. For example, professionals working in dynamic environments or those with accessibility needs might find this interface particularly useful. The textual interface, while less flexible in dynamic

contexts, was appreciated for its precision and suitability in traditional coding environments.
- Preference and Usability: Despite the potential of the Multimodal Extension, the majority of participants expressed a preference for the traditional interface for most tasks due to its familiarity and reliability. However, some found the multimodal interface more engaging and enjoyable for creative tasks, suggesting that it could complement rather than replace traditional methods.
- Improvements Needed: Feedback indicated that the gesture recognition accuracy and voice command responsiveness require significant improvements for the multimodal interface to be viable for daily use.Participants highlighted the need for better error correction mechanisms, more intuitive gesture sets, and the possibility was considered that some issues might be caused by occasional erroneous responses from the Language Model (LLM). Enhancements in these areas could make the interface more user-friendly and reduce the occurrence of errors.
- Task Suitability: Participants noted that simple tasks were easier and faster with the Visual Extension, while more complex tasks involving multiple regions and dynamic interactions could benefit from the multimodal interface once the initial learning curve is overcome. This indicates that each interface has strengths in different areas of task complexity.

5.4 Participant Preferences

When asked to choose their preferred tool for creating and editing regions, 5 out of 7 respondents indicated a preference for the Visual Extension. The primary reasons for this preference included greater accuracy, ease of use, and reliability. Conversely, the remaining participants who favored the gesture and voice-based interface highlighted the novelty of the interaction and the potential for more natural and intuitive controls once the learning curve is overcome. Comments such as "the multimodal interface feels more futuristic and engaging" were common among this group, reflecting their enthusiasm for the innovative approach.

5.5 Additional Insights

The study also revealed that the participants experience levels significantly influenced their preferences and performance. Advanced users were more likely to appreciate the efficiency gains from the multimodal interface, while beginners found the Visual Extension more accessible and straightforward. This suggests that the Multimodal Extension could be more beneficial as an auxiliary tool for experienced developers rather than a complete replacement for traditional methods.

5.6 Detailed Task Analysis

Further analysis of specific tasks showed that:

- Region Creation: Participants found that creating new regions was more intuitive with the multimodal interface once they became proficient, thanks to the ability to use voice commands for specifying dimensions and positions. It can be inferred that initially the visual version brings greater satisfaction, but after learning how to use the tool, the multimodal extension can be very useful.
- Region Editing: Editing existing regions, such as resizing and moving, was more accurate with the visual interface. The precision required for these tasks often led to frustration with gesture-based controls.
- Complex Interactions: Tasks involving multiple regions and dynamic adjustments highlighted the potential of the multimodal interface. Participants who mastered the gestures and voice commands found these tasks, such as creating multiple regions in a specific layout, easier and more efficient compared to traditional methods.

5.7 Interview Feedback

In addition to the quantitative data collected, we conducted interviews with participants who accepted an interview to obtain deeper feedback on their experiences and preferences. One participant provided particularly insightful feedback about the suitability of each interface for different types of tasks.

The participant reported that for simpler applications, the Visual Extension is more useful due to its straightforward and familiar nature. They felt that the traditional interface allows for quick and precise interactions without the need for extensive adjustments or corrections. This preference aligns with the higher ease of use and accuracy ratings observed in the quantitative data. Another interviewee highlighted the ease of using the Visual Extension to quickly understand where their content would be displayed, emphasizing its effectiveness for straightforward visualization tasks.

However, the participant noted a significant advantage of the multimodal interface for more complex tasks that involve managing many regions. They highlighted that the ability to use voice commands and gestures for creating and manipulating regions made these tasks less cumbersome and more efficient. Specifically, for tasks requiring multiple adjustments and dynamic interactions, the multimodal interface's intuitive control mechanisms reduced the cognitive load and made the process more engaging.

This feedback underscores the potential for a hybrid approach where the traditional interface could be used for simpler, more precise tasks, while the multimodal interface could be leveraged for handling complexity and enhancing user engagement in more demanding scenarios. It suggests that offering users the flexibility to switch between interfaces based on the task at hand could maximize productivity and satisfaction.

The participant's insights reinforce the need for further development and refinement of the multimodal interface to fully realize its potential for complex interactions while maintaining the reliability and precision of the traditional interface for simpler tasks.

Furthermore, there are threats to the validity of this study, such as the small number of participants and the variability in their experience with NCL and VS Code, which may limit the generalizability of the results. The reliance on voice and gesture recognition technologies, which are prone to errors, may also have influenced users perceptions of the effectiveness of the Multimodal Extension. Therefore, future research should consider a larger and more homogeneous sample of users and improvements in the supporting technologies to ensure more robust and representative results.

6 CONCLUSION

This paper presents two approaches to facilitate the creation and manipulation of NCL regions graphically. The first approach uses traditional mouse and keyboard interfaces, while the second integrates gesture recognition and voice commands into a VS Code extension to enhance the editing experience for NCL documents. We experimented to evaluate the usability and developers' preferences for both tools. Our initial results indicate that, while the traditional mouse and keyboard interface remains superior in terms of user satisfaction and perceived accuracy, the multimodal interface shows promise by significantly reducing task completion time and the number of errors.

The integration of these new tools that help author NCL documents visually into VS Code for editing can represent a step towards more practical and easy-to-use development environments. While there are challenges to address, the potential benefits in terms of efficiency, availability, and user satisfaction are substantial. By focusing on continuous improvement and user-centred design, we anticipate that this approach can positively influence the future development of multimedia applications

Future work will focus on refining the tasks performed using the multimodal interface to ensure they match or exceed the efficiency and accuracy of manual text editing. This includes improving gesture accuracy, enhancing the responsiveness of voice commands, and expanding the range of supported editing actions. Understanding the potential of using LLMs to generate code snippets, we believe a multi-agent approach could efficiently resolve multiple blocks within an NCL document. By providing more comprehensive support for generating entire NCL code blocks, we aim to offer a tool that is not only innovative but also practical for everyday use by developers.

Furthermore, exploring the integration of artificial intelligence and machine learning techniques to enhance the accuracy of gesture and voice recognition could lead to even more robust and intelligent systems. AI-driven predictive models could anticipate user actions and offer contextual assistance, thereby improving overall efficiency. Future work could explore the possibility of allowing users to select from various Large Language Models (LLMs) based on their specific needs or preferences. This customization would enable users to optimize the tool's performance for different tasks or domains, enhancing the overall flexibility and utility of the system. Additionally, enabling users to submit their LLMs could extend the tool's capabilities, allowing for tailored solutions that leverage specialized knowledge or proprietary data.

Finally, conducting extensive user studies with a larger and more diverse participant pool will be crucial. This will help validate the findings of our initial experiments and ensure that the multimodal interface meets the needs of a broad range of users. Gathering detailed feedback on usability, effectiveness, and user satisfaction will provide valuable insights that can guide the continuous improvement of the tool.

ETHICS STATEMENT

In this work, we adhere to ethical guidelines throughout our experiments involving volunteers. All participants were provided with an Informed Consent Form before their involvement, ensuring they were fully informed about the research scope, procedures, and their rights. We ensure the protection of participants' data by exclusively using non-sensitive anonymized data, minimizing risks in data handling.

In addition, a comprehensive document detailing all aspects of the research, including objectives, methodology, potential risks, and benefits, was submitted to the Ethics Committee of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) for review and approval, demonstrating our commitment to ethical research practices.

ACKNOWLEDGMENTS

The authors would like to acknowledge RNP (Rede Nacional de Ensino e Pesquisa) and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support.

REFERENCES

[1] NBR ABNT. [n. d.]. Digital Terrestrial Television-Data Coding and Transmission Specification for Digital Broadcasting-Part 2: Ginga-NCL for fixed and mobile receivers, Brazilian Standard 15606-2, Brazil, 2007.
[2] G Kumar Arora. 2017. *SOLID Principles Succinctly.* CreateSpace Independent Publishing Platform. 1-4.
[3] Abel Avram. 2007. *Domain-driven design Quickly. 20-32.* Lulu.com.
[4] Aaron Bangor, Philip T Kortum, and James T Miller. 2008. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.
[5] Moniruzzaman Bhuiyan and Rich Picking. 2009. Gesture-controlled user interfaces, what have we done and what's next. In *Proceedings of the fifth collaborative research symposium on security, E-Learning, Internet and Networking (SEIN 2009), Darmstadt, Germany.* Citeseer, 26–27.
[6] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).
[8] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128* (2023).
[9] Visual Studio Code. 2019. Visual studio code. *Recuperado el Octubre de* (2019).
[10] Douglas Paulo de Mattos and Débora C Muchaluat-Saade. 2018. Steve: A hypermedia authoring tool based on the simple interactive multimedia model. In *Proceedings of the ACM Symposium on Document Engineering 2018.* 1–10.
[11] Joel André Ferreira Dos Santos and Débora Christina Muchaluat-Saade. 2012. XTemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions. *Multimedia Tools and Applications* 61, 3 (2012), 645–673.
[12] Louie Giray. 2023. Prompt engineering with ChatGPT: a guide for academic writers. *Annals of biomedical engineering* 51, 12 (2023), 2629–2633.
[13] Moh Harris, Ali Suryaperdana Agoes, et al. 2021. Applying hand gesture recognition for user guide application using MediaPipe. In *2nd International Seminar of Science and Applied Technology (ISSAT 2021).* Atlantis Press, 101–108.
[14] Jhilmil Jain, Arnold Lund, and Dennis Wixon. 2011. The future of natural user interfaces. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems.* 211–214.
[15] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. 2015. Likert scale: Explored and explained. *British journal of applied science & technology* 7, 4 (2015), 396–403.
[16] Bipin Joshi and Bipin Joshi. 2016. Overview of SOLID Principles and Design Patterns. *Beginning SOLID Principles and Design Patterns for ASP. NET Developers* (2016), 1–44.
[17] Dr Manju Kaushik and Rashmi Jain. 2014. Natural user interfaces: Trend in virtual interaction. *arXiv preprint arXiv:1405.0101* (2014).
[18] James R Lewis. 2018. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction* 34, 7 (2018), 577–590.
[19] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science* 378, 6624 (2022), 1092–1097.
[20] Scott Millett and Nick Tune. 2015. *Patterns, principles, and practices of domain-driven design. 50-64.* John Wiley & Sons.
[21] Daniel de Sousa Moraes, Polyana Bezerra da Costa, Antonio JG Busson, José Matheus Carvalho Boaro, Carlos de Salles Soares Neto, and Sergio Colcher. 2023.

On the Challenges of Using Large Language Models for NCL Code Generation. In *Anais Estendidos do XXIX Simpósio Brasileiro de Sistemas Multimídia e Web*. SBC, 151–156.

[22] Daniel de Sousa Moraes, André Luiz de B Damasceno, Antonio José G Busson, and Carlos de Salles Soares Neto. 2016. Lua2NCL: framework for textual authoring of NCL applications using Lua. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. 47–54.

[23] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474* (2022).

[24] Douglas Paulo de Mattos, Júlia Varanda da Silva, and Débora Christina Muchaluat-Saade. 2013. NEXT: graphical editor for authoring NCL documents supporting composite templates. In *Proceedings of the 11th european conference on Interactive TV and video*. 89–98.

[25] Carlos de Salles Soares Neto Roberto Gerson de Albuquerque Azevedo, Mario Meireles Teixeira. 2009. NCL Eclipse: Ambiente Integrado para o Desenvolvimento de Aplicações para TV Digital Interativa em Nested Context Language. In *Salão de Ferramentas - SBRC 2009*. São Luís, MA, Brazil.

[26] Victor Hazin da Rocha. 2013. *DiTV–Arquitetura de desenvolvimento para aplicações interativas distribuídas para TV digital.* Master's thesis. Universidade Federal de Pernambuco.

[27] Harmeet Singh and Syed Imtiyaz Hassan. 2015. Effect of solid design principles on quality of software: An empirical assessment. *International Journal of Scientific & Engineering Research* 6, 4 (2015), 1321–1324.

[28] Luiz Fernando Gomes Soares and Rogério Ferreira Rodrigues. 2006. Nested context language 3.0 part 8–ncl digital tv profiles. *Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio* 1200, 35 (2006), 06.

[29] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382* (2023).

[30] Anthony Zhang. 2017. SpeechRecognition 2.1.3. https://pypi.org/project/SpeechRecognition/2.1.3/. Accessed: 2024-08-20.

[31] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. 2020. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214* (2020).