Interfaces for Alternatives, Difference Visualizations and Collaborative Authoring of Behavior Trees

by

Md. Yousuf Hossain

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Science in Computer Science

Faculty of Business and Information Technology

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

December 2020

THESIS EXAMINATION INFORMATION

Submitted by: **Md. Yousuf Hossain**

Master of Science in **Computer Science**

Thesis title: Interfaces for Alternatives, Difference Visualizations and Collaborative Authoring of Behavior Trees

An oral defense of this thesis took place on December 2, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Faisal Qureshi
Research Supervisor	Dr. Loutfouz Zaman
Examining Committee Member	Dr. Alvaro Joffre Uribe Quevedo
Thesis Examiner	Dr. Bill Kapralos

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

A visual behavior authoring framework for exploring and comparing multiple game AI behavior alternatives is presented. A mixed method study revealed participants rated it higher compared to the control framework. Mostly positive feedback was also obtained in a semi-structured interview and a follow-up qualitative longitudinal study. An extension for collaboration using synchronous and asynchronous modes is also presented. A mixed methods study revealed that collaboration provides improvement over the control condition, but no differences were found between the collaboration modes. In a semi-structured interview, the synchronous mode was revealed to be useful for quick prototyping, while the asynchronous mode – for most other situations. Overall, the results suggest that proposed solutions have a potential to improve the exploration and collaboration for behavior tree development, but further research is necessary.

AUTHOR'S DECLARATION

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

The research work in this thesis that was performed in compliance with the regulations of Research Ethics Board/Animal Care Committee under Research Ethics Board 16071 and 15463.

Md. Yousuf Hossain

STATEMENT OF CONTRIBUTIONS

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

ACKNOWLEDGEMENTS

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research supervisor Dr. Loutfouz Zaman. I will always grateful for the opportunity that you gave me and the time and effort you have put towards my research.

Last but not the least, I would like to thank my parents: My mother (The love of my life) and my father for always supporting me spiritually throughout my life.

TABLE OF CONTENTS

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Behavior Trees and Alternatives

A *behavior tree* (BT) is a graphical modeling language to create artificial intelligence (AI) behavior in video games, visual simulators, robotics, among others. BTs have been used in video games since the mid-2000s [63] (p.349). Although other game AI technologies currently exist, such as *Utility AI* [95] for implementing complex non-player character (NPC) behavior, BTs today remain popular and continue to be included in game engines, e.g., *Unreal Engine* [111]. Due to their simplicity of use, BTs make developing AI behavior accessible even to non-programmers [61], which is arguably one of the reasons behind their continuing popularity.

NodeCanvas [82] is a visual behavior authoring framework for *Unity* [109], which enables developers to create advanced AI behaviors and logic. Although *NodeCanvas* and other BT systems make game development more accessible for developers with limited programming skills, arguably these systems still do not fully support development of *conceptual models.* A conceptual model is "*an outline of what people can do with a product and which concepts are needed for the user to understand how to interact with it*" [98]. Arguably, this is also important in game development: Every game starts as an idea, which the developers add on to iteratively. During this process, game developers frequently explore various possible solutions before they decide on the preferred option. We refer to these possible options as *alternatives*. Designing alternatives is one of the four core activities of design and is part of the development phase of the *double diamond of design* [98]. Using BT editors, the exploration of alternatives is possible using traditional methods such as duplication,

reconnecting nodes and conditional execution. This is limiting because repetition, elimination and conditional execution come with a risk of losing prior progress. Here we present a solution which enables the exploration of alternative AI behaviors in the development cycle in a streamlined manner to improve the workflow of the game developers.

Designing prototypes is another important part of the development phase of the double diamond of design [98]. Since the interaction design involves designing the behavior of interactive products in a short period, an effective way for users to evaluate such designs is to interact with them, which can be achieved with prototyping [98]. The use of BTs for prototyping has been previously identified, see e.g., [25, 58]. *NCAlt* can also facilitate the development cycle by streamlining prototyping through the support of alternatives and difference visualizations.

Methods have been proposed to bring the support for alternatives to vector graphics [65], generative design [121, 120, 119] and, more recently, to visual game logic scripting [71]. To the best of our knowledge, we are the first to try alternatives specifically for AI development to test multiple the AI agents– a domain which affords for a unique opportunity for real-time, simultaneous and interactive exploration of multiple behaviors in the same game scene. We implemented this solution in a system we called *NCAlt* (*NodeCanvas* Alternatives), which is an extension to *NodeCanvas* – a node-based behavior authoring framework to create BTs for game AI. *NCAlt* supports multiple BT alternatives for a game character and allows real-time differencing of multiple behaviors in both the game scene and the tree visualization.

The first contribution of this thesis includes a system that enables:

- creating and managing alternatives for BTs;

- comparing BTs using a one-to-many difference visualization technique;
- real-time play testing of multiple NPC behavior alternatives simultaneously using color coding and labeling;

Additionally, the system has been evaluated with:

- a user study which compares the above to *NodeCanvas* for fixed tasks;
- a longitudinal study on the system for an open-ended task.

1.2 Collaboration

Today video game development is a collective process in which a variety of professionals from different backgrounds, such as game designers, programmers, QA specialists, come together and collaborate on a common task by constantly communicating with each other. Solutions for the support of distributed collaboration such as project management [4, 87, 94, 103], version control [39], instant messaging [26, 114], and video conferencing [43, 113] currently facilitate this. Being stand-alone, these systems are not aligned with the programming environment, which may result in either substantial distraction or their underuse due to flow interruption.

People are inherently social, requiring to collaborate, coordinate, and communicate with one another, and the diverse range of applications has emerged to enable them to do so in extensive and diverse ways [98]. People often learn effectively when collaborating together, in particular with the help of technology [98]. *Pair programming* is an agile technique in which two developers work together on the same code or project. Pair programming was found beneficial for teaching and engaging learners, see e.g., [101, 117]. Arguably there may exist an increased and urgent need for tools that support collaboration during the pandemic, when the only option to learn for

many remains through distance learning. While work has been done for code-based programming, visual pair programming in comparison has not received as much attention for collaboration until now.

A visual programming language is a graphical programming language [97, 122] where the users create programs by manipulating program elements graphically rather than by writing textual code [60]. Due to its simplicity of use and low learning curve, visual programming has gained popularity among non-programmers for developing game AI [61]. Using node-based programming, applications are developed by connecting "blocks" of self-contained code. Node-based and other visual programming interfaces have become commonplace in popular game development environments such as *Unreal Engine* and *Unity*, as they facilitate and expedite prototyping and development. Collaborative options in these tools are also available. Work has been done on collaboration for source code [55, 57, 101], and more recently, for visual programming in education with *Blockly* [112]. However, to the best of our knowledge, currently there is no collaborative environment available for behavior trees. In our work we aim to demonstrate that a collaborative visual programming environment for developing game AI is a viable approach. In our work, we focus specifically on the development process of collaborative game AI using behavior trees, as we believe it can help to close the gap between the programmers and non-programmers through the use of visual pair programming and also to facilitate distance learning. As our work will demonstrate this also has a potential to facilitate the process of prototyping.

As a solution, we built *NCCollab* (*NodeCanvas Collaboration*), which is an extension to *NodeCanvas* [81] – a node-based behavior authoring framework to create behavior trees (BTs) for game AI. *NCCollab* facilitates collaborative visual programming of game AI between multiple developers. *NCCollab* supports two modes

of collaboration: *synchronous* and *asynchronous*. In the synchronous mode, multiple users work on the same BT, where the changes are merged between these users in real time. In the asynchronous mode, multiple users can work on the same or different BT and they can merge their changes with other users' changes whenever they are ready. *NCCollab* is integrated with *NCAlt* (*NodeCanvas Alternatives*), which is another extension to *NodeCanvas* we built earlier to create, merge, and manage multiple versions of BT alternatives and compare their behaviors.

The second contribution of this thesis features *NCCollab* – a node-based behavior authoring framework to create BTs, and include:

- synchronous and asynchronous collaboration modes;
- live preview of other collaborators' BT alternatives;
- BT history, which allows restoring previous states from history;
- a semi-automatic conflict resolution interface to resolve conflicts that may arise during collaboration;
- an improved one-to-many difference visualization technique which was adapted from *NCAlt* for showing differences between collaborators' BT alternatives;
- an instant messaging system for collaboration, and
- a user study which compares synchronous and asynchronous collaboration to a control condition using fixed tasks.

Chapter 2
Related Work

2.1 Earlier Work on UIs for Alternatives

Marks et al. [73] presented *Design Gallery*, which was an early effort to show multiple variations of 3D graphics and animations in a single view. Alternatives were introduced in a wide range of applications by Lunzer et al. for the comparison of queries over a multi-attribute set of data, [68, 70], the collection and comparison of results from multiple alternative resources offering nominally the same processing [66, 67], online learning [59] and for accessing information, real-time simulation, and document design [69]. Terry et al. presented several techniques to support the exploration of variations and continuous analysis of someone's work, such as *design horizon* [104], an interface to support multiple alternatives for screenshots. Thereafter, Terry et al. also presented another work, *Parallel Paths* [105], an interaction model that generates image alternatives, as well as side-by-side comparison of multiple alternative solutions.

NCAlt also facilitates creation, management and comparison of multiple alternatives in a single workspace. It draws upon the early ideas stated above and adapts them for live interactions with multiple game AI behaviors simultaneously.

2.2 Alternatives for Parametric and Node-Based Design Systems

The origins of our work are positioned in the works on alternatives in parametric and node-based design systems. Parametric Computer-Aided Design (PCAD) is a domain that uses the computer to design different structures or systems that represent real-world component properties [89]. Node-based interfaces further empower PCAD by incorporating visual scripting. The use of alternatives with these technologies has experienced an increased popularity in the past decade. Exploration often involves

specifying different values for a series of parameters and comparing the outcomes in parametric systems. Shireen et al. explored a generative design tool [99] that used large displays for exploring numerous design options. Kolarić et al. introduced *CAMBRIA* [64], a multi-state prototyping tool for simultaneously managing multiple 2D vector graphics models. Cristie and Joyce introduced two web-based platforms *GHShot* [23] and a plugin [22] for *Grasshopper* [45] that allow designers to collaborate parametric structures in a similar way to *Git* [39] and *Github* [13] with branching support and allowing users to save the current design state as a 3D model and parametric information together. *GHShot* was implemented in an architecture class for working in a group and to provide feedback to each other's design. Woodbury et al. [116] introduced a prototype gallery system on a web browser, which supports saving alternatives into the gallery, sharing them with others, and merging them to generate more alternatives. Mohiuddin et al. [75] outlined a special type of collection called the *Parallel Coordinate View* to accommodate multiple references to the same alternative or collection in an interface. In this work, an online gallery system was presented for design alternatives in parametric modeling to support multiple commercially available parametric modelers. Guenther introduced *Shiro* [46] – a declarative language which enables representing multiple alternatives in a single parametric system. The language was designed for use with parametric and node-based systems. Elkhaldi and Woodbury [32] presented *Alt.Text* – a node-based text document construction platform that promotes alternative production through a hierarchical multi-state database model. The tool features a user interface to view alternatives and edit text simultaneously.

Most of the works stated above lacked rigorous formal evaluation. However, Zaman et al. evaluated *GEM-NI* [121] – a node-based generative design tool that allows the user to create alternatives by generating branches, merging with other *alternatives*

and retrieving history, among others. The support for alternatives in this study was well received. Alternatives at graph and node levels were introduced for *Unreal Blueprints* [9] in *SuBVis* [71], which received positive feedback in a study with four participants.

NCAlt builds upon previous work on node-based systems on creating multiple alternatives [64, 71, 116, 121], difference visualization [120] and merging [121] and adapts these interfaces to BTs.

2.3 Alternatives and Related Concepts in Other Domains

The use of alternatives is not unique to design software. Nancel and Cockburn presented *Causality* [80] – a conceptual interaction model that keeps track of previous states to be able to edit from history. They used this tool to show a use case scenario for a painting application. Hartmann et al. introduced *Juxtapose* [54] – a code editor and runtime environment for creating multiple design alternatives. They introduced a new technique for rapidly comparing code alternatives by selective parallel source editing and execution. Moreover, it automatically creates control interfaces to explore parameter variations during runtime. The authors found in a user study that designers can leverage these two techniques to survey more options and faster. Hailpern et al. [51] evaluated *Team Storm* – a system that allows collaboration of multiple design ideas in parallel and found that design teams are able to effectively utilize the tool to create, organize, and share multiple design ideas during creative group work. Smith et al. [100] presented a computational sketching tool and ran a user study to see the effects of three interaction models: tab interfaces, layered canvases, and spatial maps while working with multiple alternatives in early design stages. They found all three interaction models to be useful. Kazi et al. [62] presented *DreamSketch* – a 3D design interface where a user can roughly sketch the design context, and a generative design algorithm automatically

shows multiple design alternatives. *The What-If Tool* [115] has been used for probing, visualizing and analyzing machine learning system with minimal coding. Although not an alternatives system, *LevelMerge* [96], is a collaborative environment for game level editing which supports editing of scenes and behavior scripts in *Unity*. *LevelMerge* also supports diffing and merging, which is related to *GEM-NI* [120, 121]. *LevelMerge*, however, has not been formally evaluated.

With the exception of *Juxtapose* [54] and *LevelMerge* [96], the works mentioned above are orthogonal to ours. However, they underline the continuing *interest* in alternatives beyond PCAD.

2.4 Difference Visualizations of Trees

Graham et al. [44] surveyed different representation techniques for a single tree, trees in pair and multi-tree visualization. Furthermore, Gleicher et al. [40] proposed a general taxonomy of visual designs that can be used for comparison of three basic categories of trees. In these two works the following common techniques were identified: juxtaposition (e.g., [77]), overlay (e.g., [24, 118]) , explicit codes (e.g., [120]), animation ([15], [56]), matrix, agglomeration and edge drawing.

Side-by-side views is a technique related to our variant of one-to-many diffing. Examples of this technique include *DualNet* [78], which visualizes sub-networks of node-link diagrams with side-by-side views. *TreeJuxtaposer* [77] compares large trees with side-by-side views. *TreeVersity* [47–50] shows changes in topology and node values by using glyphs that pre-attentively highlight changes. *TreeVersity* also highlights new and deleted nodes. Bremm et al. [11] introduced a new technique for comparing multiple hierarchies on global and local structures for use in biology. This work is orthogonal since it targets trees that are significantly more complex than

behavior trees. Moreover, our trees encode priority information, which cannot be captured using this approach. *MACE* [120] is an interface for comparing two or more generative node-based model alternatives using a one-to-many diffing interface. In this work, two types of encodings were introduced for diffing in the model viewer and both were found effective.

From the above, *NCAlt* employs explicit codes, a variant of change highlights, juxtaposition and a variant of overlay for showing deleted nodes. *NCAlt* also builds on the additive encoding and a one-to-many diffing technique of *MACE* [120], by introducing an additional encoding for priority change, which is unique to BTs. *NCAlt* uses the top-down layout as it is an easier layout for visualizing common BT systems in game development tools, such as *Unreal Engine's Behavior Trees* [6], *Behavior Designer* [5], etc.

2.5 Collaboration in Programming Environments

The use of collaboration is a well-established practice in a variety of fields, including node-based programming. Goldman et al. [41] introduced *Collabode*– a web-based Java integrated development environment designed to support close, synchronous collaboration between multiple programmers. Goldman et al. [41] examined the problem of collaborative coding in the context of program compilation errors introduced by other users which make collaboration more difficult and describe an algorithm for error mediated integration of program code. They evaluated this collaboration algorithm and interface on recorded data from previous pilot experiments with *Collabode*, and via a user study with student and professional programmers. They found that *Collabode* offers appreciable benefits over naïve continuous synchronization without regard to errors and over manual version control. *NCCollab* does not directly stop errors from

happening and does not try to solve them. However, *NCCollab* makes sure if there is an error in the tree, the updates are not sent to the server to prevent other users from becoming affected by the error. Ghorashi et al. [101] created *Jimbo*, a collaborative IDE with live preview with an intention of facilitating distance learning of programming. *Jimbo* integrates text chat, audio discussion, inline discussion and live preview of programming to support better collaboration, communication, and bridge gaps and develop mutual understanding between designers and developers. *Jimbo* [101] lacks a formal study to test the collaborative features in a real scenario. To facilitate better communications, *NCCollab* also aims to facilitate this for game developers through the use of instant messaging, live preview of other users' BT, restoring BTs from history, conflict resolution, and by repurposing vanilla *NodeCanvas* features such as, inline commenting through node comments and creating canvas groups. Besides, *NCCollab* was formally evaluated with a control condition using fixed tasks. Berland et al. [8] introduced *AMOEBA* – a tool to support the instrumentation of communication between novice programmers in non-traditional programming environments. Traditional programming approaches involve a programmer working individually manipulating a single piece of code. However, *AMOEBA* was developed and utilized to facilitate collaboration in a learning scenario. Berland et al. [8] ran a formal study in a classroom setting between inexperienced middle school and high school programmers using the Intelligent Programming (IPRO) framework. Nosek et al. [84] introduced a collaborative programming environment where two programmers working jointly on the same algorithm and code. Nosek et al. [84] performed a comparative study, which found that student programmers working collaboratively outperformed individual programmers for aspects of performance, which included readability, functionality, and time and satisfaction (as confidence and enjoyment). Fan et al. [33] introduced a novel

Any-Time Collaborative Programming Environment (*ATCoPE*) to combine traditional non-real-time collaborative programming tools and environments seamlessly with emerging collaborative real-time programming techniques. *ATCoPE* enables collaborative programmers to work in software development scenarios and move flexibly according to their needs between different collaboration modes. Fan et al. [33] have proposed a functional design for *ATCoPE*. However, although they validated the feasibility performance evaluation, they did not implement the system and did not run a formal user study. Examples in education include *Blockly* [112] and *Scratch* [97], which use visual programming for collaborative developing 2D interactive applications.

The works above feature both synchronous and asynchronous collaborative systems for code-based programming. As a result, visual programming specific techniques for supporting collaboration are not present in this work. *NCCollab*, in contrast, features access to the history, the ability to revert to previous states, or graphical difference visualizations between collaborators' digital works. *NCCollab* also offers a conflict resolution for solving conflicts during collaboration, which is quite different from a traditional programming language because tradition programming languages are script based but *NCCollab* is tree-based language. Although underneath the tree-based structure there is Json file to store all the information for the BT, the users never interact with that Json file directly. Since the traditional conflict resolutions algorithms are built for code-based programming language, there is very little to no use of them in regards to a visual scripting too such as *NCCollab*.

2.6 Collaboration in Non-programming Environments

Research on collaboration is an extensive field. Here, we will only focus on works that support virtual/remote collaboration. The goal of *NCCollab* is to support collaborative

game AI authoring in the style of cloud-based services such as Google *Docs* [42], Microsoft *OneDrive* [74], *Git* [39], Expat Software *Twiddla* [108], Nulab [85], and *Cacoo* [86] because these are the common platforms used for collaboration. These platforms allow addition of files in *Git*, texts in Google *Docs* and *OneDrive*, shapes in *Twiddla* and *Cacco* from multiple users with no conflicts or warnings.

Operations	***Google Doc*** **(Syn)**	***OneDrive*** **(Asyn)**	**Git** **(Asyn)**	***Twiddla*** **(Syn)**	***Cacoo*** **(Syn)**
Add in different Paragraph (Text)/ Branch (Tree)/ Shape (Diagram)	No warning	No warning	No warning	No warning	No warning
Add in same Paragraph (Text)/ Branch (Tree)/ Shape (Diagram)	No warning	Paragraph locked while editing.	No warning	No warning	Wait until shape completed, No warning
Update/Overwrite	No warning	Paragraph locked while editing.	Warning to manually fix conflict	No warning	Wait until shape completed, No warning
Delete	No warning	Paragraph locked while editing.	Force Delete	No warning	No warning

Table 2-1. A comparison of collaborative tools for common operations.

Table 2-1 presents a comparison between these collaborative tools for operations such as addition, deletion and updating. Flowchart or text-based tools such as *Twiddla*, *Nulab*, and *Cacoo* allow a user to delete or update another user's file without warning. However, version control systems such as *Git* always ensures a user does not delete or

modify a file belonging to another user by mistake. For example, a user has pushed their modified file to a remote server while other users are still modifying previous version of the file before the first user pushes changes. In this case, these other users have inconsistent versions of the file with compared to the remote server's version. Now, if they try to push their versions, *Git* will not allow this to prevent conflicts. In *NCCollab* only addition of nodes is allowed without warnings but, when it comes to deletion or updating, the user needs to solve every conflict to merge with another user's BT. Also, in *NCCollab*, similar to *Git*, if the user deletes or makes changes during collaboration that they are not satisfied with, there is an option to restore a previous version of the BT from the history.

2.7 Collaboration in Game Development

Santoni et al. [96] presented *LevelMerge* − a collaborative game level editing tool that supports editing scenes and behavior scripts in *Unity* by merging labeled graphs. Using *LevelMerge*, level designers can be designing the game scene, artists can be designing 3D models or its materials, while programmers can work on the scripts at the same time. The system, however, has not been formally evaluated. In contrast, *NCCollab* focuses on a more specific aspect of game development collaboration which is developing game AI. Similar to *LevelMerge* [96], the intrinsic hierarchical structure, graphs are commonly used to represent game levels. For example, Doboš et al. [28] presented *XML3DRepo* − an optimized API for game scenes with versioning and encoding all the objects and assets into a unified game scene graph. The system has been evaluated on the basis of the cumulative CPU decoding time and the overall download time for a variety of 3D model by the authors. However, the system was not evaluated by the research participants.

The industry and academic tools mentioned here allow collaboration between users of different backgrounds. NCCollab was designed to support collaboration between programmers or AI designers only. NCCollab can also facilitate collaboration between programmers and non-programmers, since one of the key benefits of visual scripting tools is that it can be understood by users with limited programming experience.

2.8 Versioning

NCCollab implements tracking history, live preview, and resolution of collaboration conflicts for BTs. These are the features commonly found in version control [28, 39]. Traditionally these techniques have been used for source code and text but, more recently, they have been adapted for more sophisticated types of data. Methods that utilize recording of editing operations have been introduced by Chen et al. [17] for 2D images and by Chirigati et al. [20] for 3D models. In contrast, Carra and Pellacini [16] introduced a method that uses the results of editing operations for 3D scenes. Earlier examples include works by Doboš et al [27–30]. All these works, however, are orthogonal to our work, since we focus on interfaces for AI asset creation. In contrast, branching, merging, history for node-based generative design has been introduced in *GEM-NI* [121]. All except merging have been found useful in the evaluation. In addition to the collaboration features, *NCCollab* also tracks history and, therefore, can be used as a true versioning system. To the best of our knowledge we are the first to introduce a versioning interface for game AI asset creation.

Chapter 3
NodeCanvas Alternatives

Here we present *NCAlt* (*NodeCanvas Alternatives*) – a novel extension to *NodeCanvas* [81] to create, merge and manage multiple versions of BT alternatives and compare their behaviors. *NCAlt* is developed on top of *NodeCanvas* (a *Unity* engine framework) by modifying the source code of the framework. Although we are unable to share the code of *NCAlt* due to current license restrictions of *NodeCanvas* on which it builds, below we provide a detailed description of how the system works so that the reader can replicate its features. The features of *NCAlt* can be adapted in systems such as *Behavior Designer* [5], *Playmaker* [92], *Bolt* [10], *Flow Canvas* [36], and *Unreal Engine's Blueprints* [9].

Figure 3-1. A difference visualization of behavior in the game scene of the RPG game.

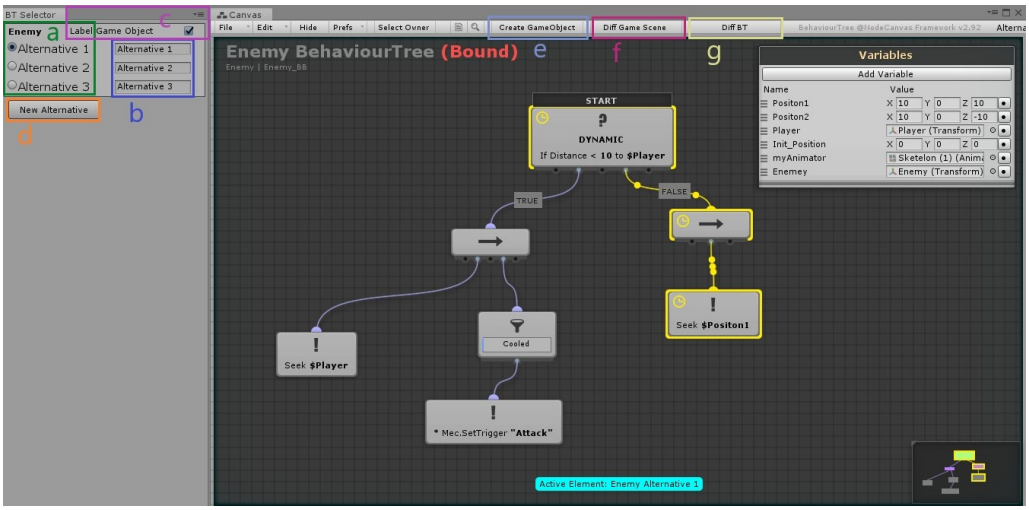


Figure 3-2. Enemy BT for Alternative 1 in the RPG game.

3.1 Multiple Behavior Tree Alternatives in NodeCanvas

In *NCAlt,* the user creates an alternative using a panel called *BT selector* (Figure 3-2). Each alternative is hosted in a dedicated editor screen. The BT selector's panel consists of:

(a) a menu widget to select the alternative,

(b) a GUI text field to rename the alternative,

(c) a Boolean toggle button to label the game object in the game scene,

(d) a GUI button to create new alternative,

(e) a GUI button to create a new clone game object with the BT alternative.

By default, alternatives are labelled incrementally as "Alternative 1", "Alternative 2", etc. When an alternative is created, *NCAlt* creates a copy of the existing BT that is being edited. This copy is saved into a separate BT JSON (JavaScript Object Notation) file.

When the user creates a new alternative, an alternative set is created in another JSON file, which consists of a unique ID and an alternative's name. The user is then

able to add and delete nodes, rewire connections, and even overwrite the entire BT alternative. However, this does not affect other BT alternatives because they are independent. We settled on this type of UI since true parallel UIs, e.g., [121], have been shown to be ineffective with the users who still preferred to work on one alternative at a time.

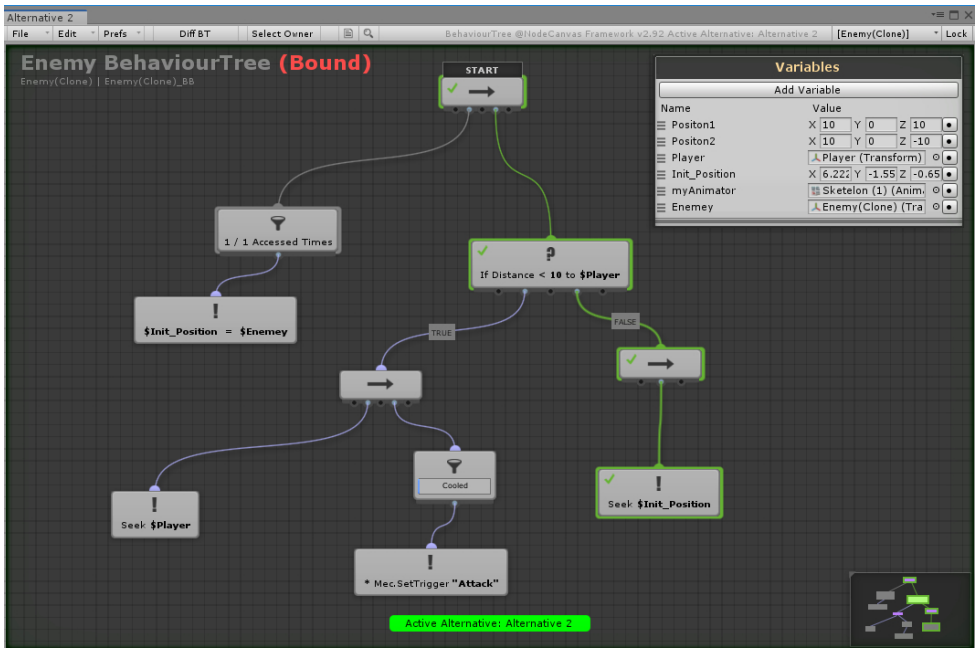


Figure 3-3. Enemy BT for Alternative 2 in the RPG game.

The following worked example features common basic RPG mechanics and is also used as a task in the user study below. Imagine John, a game developer, is trying to create an AI behavior of an enemy NPC that will follow the player when it sees them in an RPG game. If the AI finds the player character it will attack them (Figure 3-2). After creating this AI behavior, John decides to try what happens if the AI returns to its initial position when it loses the sight of that player. However, John does not want to get rid of his previous BT. So, he creates a new alternative (Figure 3-3) by pressing the "new alternative" GUI button (Figure 3-1c). This copies all the nodes from Alternative 1 to Alternative 2 and allows John to independently modify Alternative 2 without affecting Alternative 1. John adds four new nodes which are sequencer (→), 1/1 ACCESS TIMES,

$INIT_POSITION=$ENEMY, SEEK $INIT_POSITION. John then deletes $SEEK POSITION1. These AI behaviors are also observed interactively in real time in the game scene. See in Figure 3-1.

Using the radio buttons (Figure 3-1a) the user can change the active BT. This can be performed during the playtest and it allows the user to test the AI behavior of the character interactively. Furthermore, the user can label the game object in the scene view with the active BT alternative name by enabling a toggle button (Figure 3-1c). This helps the users to keep track of multiple game objects in the scene view and the currently active BT alternative for that particular game object.

In addition, when the user chooses to start the game, only the currently active combination of BTs is compiled. Therefore, there is no time penalty incurred from testing a new idea exploration.

3.2 Difference Visualizations

The differences can be visualized in both: the game view (see e.g. Figure 3-1 and Figure 3-6) and the BT (see e.g., Figure 3-8).

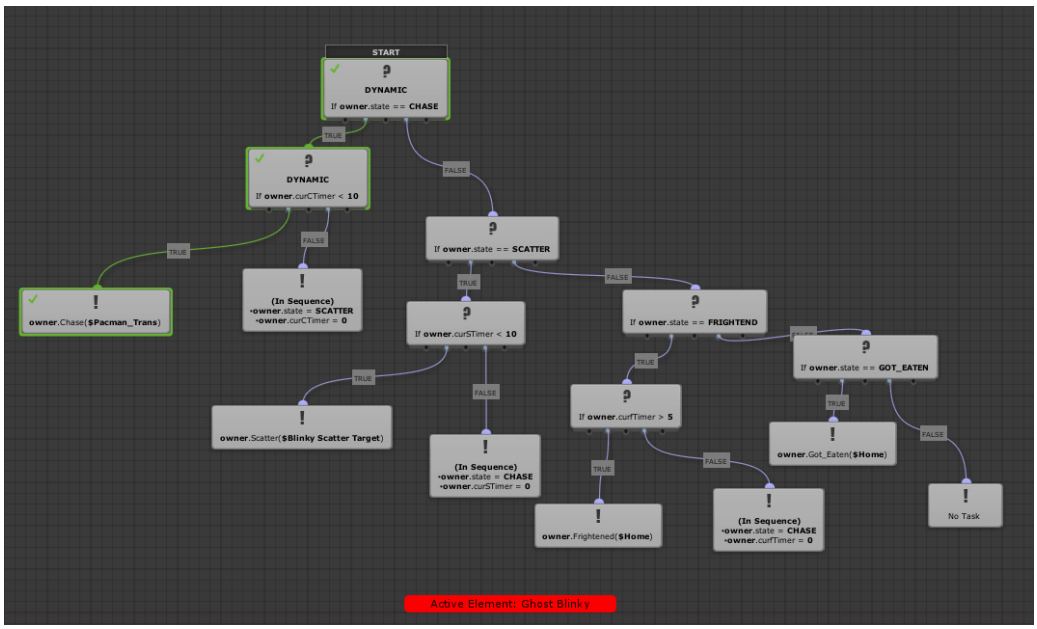
3.2.1 *Game View*

Figure 3-4. An example of *Diff Game Scene* tab.

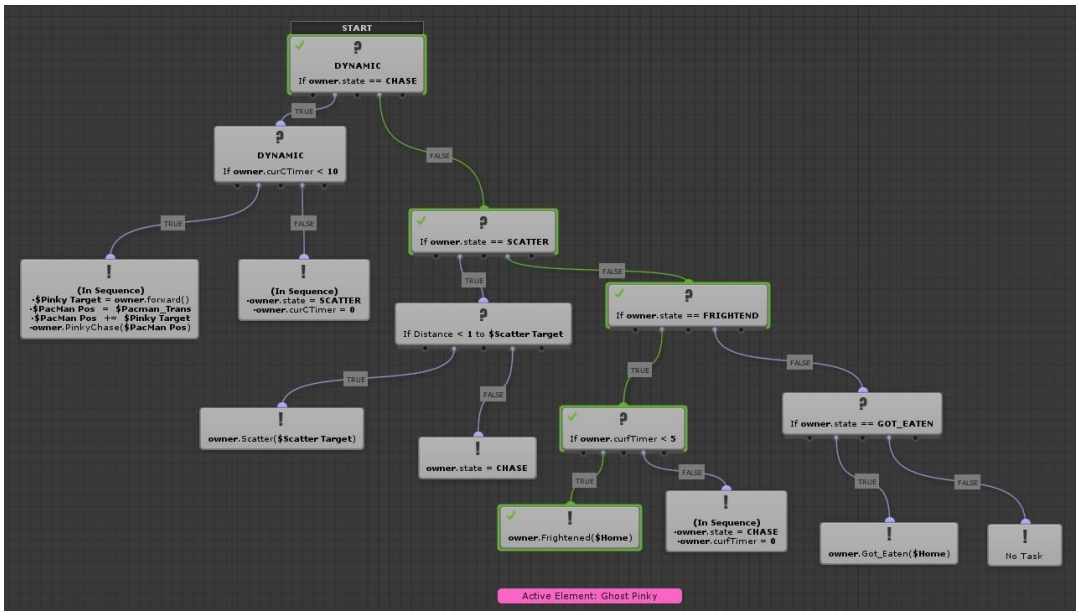
To show the differences between multiple BT alternatives in the game scene view, a clone game object is created with the compared BT alternative and colored

uniquely using the *diff game scene* toggle button color (Figure 3-4). Each compared object is labeled with the name of the alternative. To get into this mode, the user clicks on a GUI button named *diff game scene* (Figure 3-1f).

This makes a toggle button menu to appear (Figure 3-4). The user can choose any number of alternatives to compare against the active (reference) BT alternative. In the *diff game scene* tab, every toggle button is assigned a unique color for each BT in the game view. When the developer selects a toggle button from the *diff game scene* tab, a clone object is created automatically in the game scene (Figure 3-6). Another *NodeCanvas* window appears and hosts the selected BT alternative (Figure 3-5). All the selected BTs from the diff game scene tab appear in the dedicated BT canvases, which are placed together so that the user can see both the BT and the game object at the same time. To visualize every alternative uniquely in the scene they are uniquely colored and labeled with their names respectively.

(a) 

(b)



(c)

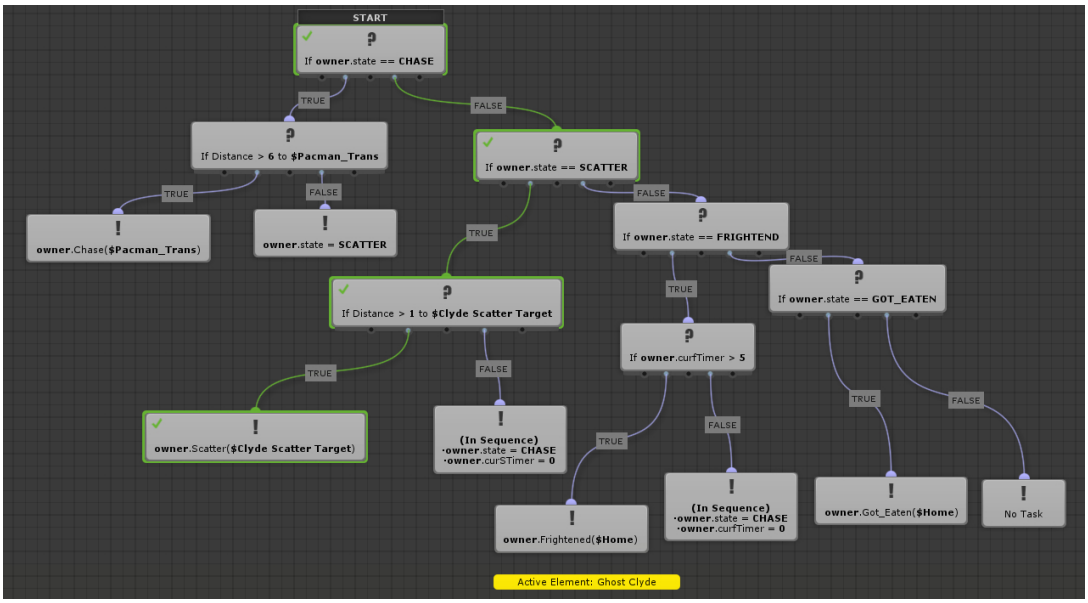


Figure 3-5. BT alternatives in *Pac-Man*: Blinky (a), Pinky (b) and Clyde's (c).

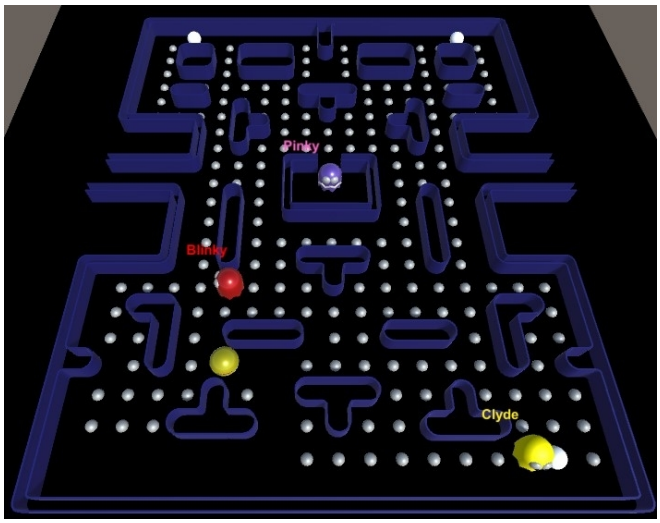


Figure 3-6. A difference visualization of behavior alternatives of Pinky, Blinky and Clyde in the game scene view of *Pac-man*.

The following worked example was designed to illustrate a simple to understand a simple use case for difference visualizations, which is also used as a task in the user study below. Imagine, Sarah a game developer, is working on developing AI behaviors for three ghosts in a game inspired by *Pac-Man* [79]. In this game, the ghosts have three different modes:

(1) *Chase mode*. All the ghosts use Pac-Man's position as a factor in selecting their target tile.

(2) *Scatter mode*. The ghost disperses to his scatter position or corner. In both the chase and scatter modes Pac-Man can be eaten by all three ghosts.

(3) *Frightened mode*. The ghost is vulnerable. He moves slower and can be eaten by Pac-Man. The ghost's color is purple. Sarah first creates a BT alternative for the ghost and names it as Blinky (red ghost in Figure 3-6). Blinky's BT is implemented in a way that he almost always follows the tile that Pac-Man currently occupies.

She then creates another ghost that will be afraid of Pac-Man by creating another BT alternative and naming it as Clyde (yellow ghost in Figure 3-6). Clyde's movements are based on how far away he currently is from Pac-Man. As soon as Clyde comes within six tiles from Pac-Man, he flees to his scatter position. Finally, Sarah decides to try one more AI for the ghost which would always be a few steps ahead of Pac-Man. She creates a new BT alternative and names it as Pinky. Pinky's target position is four tiles ahead of Pac-Man. Note in Figure 3-6 Pinky's color is purple since he is in the frightened mode. However, Pinky's original color is pink and hence the tag hovering above Pinky is pink.

As shown in Figure 3-6, Sarah created three ghost BT alternatives and now wants to see the differences in the game scene view for Blinky, Pinky, and Clyde. Blinky is the active (reference) BT alternative. Pinky and Clyde are the compared BT alternatives. Their BTs are hosted in multiple canvases and as shown in Figure 3-5. *NodeCanvas*'s active nodes and their connectors are highlighted in green.

For Blinky's BT alternative, we can see in Figure 3-5a that OWNER.STATE==CHASE, OWNER.CURCTIMER<10 and OWNER.CHASE ($PACMAN_TRANS) nodes are active. Thus, Blinky is in the chase mode (Figure 3-6). In Pinky's BT alternative (Figure 3-5b), OWNER.STATE == CHASE, OWNER.STATE == SCATTER, OWNER.STATE == FRIGHTENED, OWNER.CURFTIMER<5 and OWNER.FRIGHTEND ($HOME) nodes are active, so he is in the frightened mode moving to the home position (Figure 3-6). Clyde's BT alternative is hosted in Figure 3-5c, OWNER.STATE==CHASE, OWNER.STATE==SCATTER, DISTANCE>1 TO $SCATTER TARGET and OWNER.SACTTER ($SCATTER TARGET) nodes are active. Thus, Clyde is in the scatter mode: he is moving away from Pac-Man and going to his scatter position. See Figure 3-6.

Sarah likes Pinky's behavior and creates a new ghost game object consisting of Pinky's BT alternative only. She does this by clicking on a GUI button "Create Game Object" (Figure 3-1e), which clones a new ghost game object that only contains Pinky's BT and renames it as Pinky.

3.2.2 *One-to-Many Difference Visualizations of BTs*

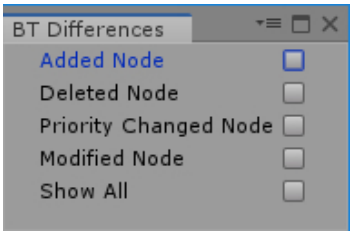
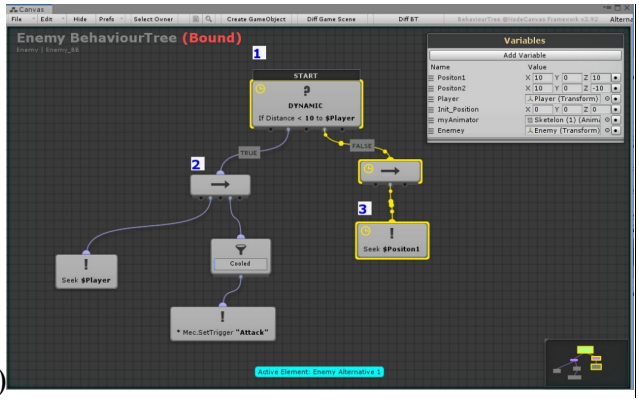
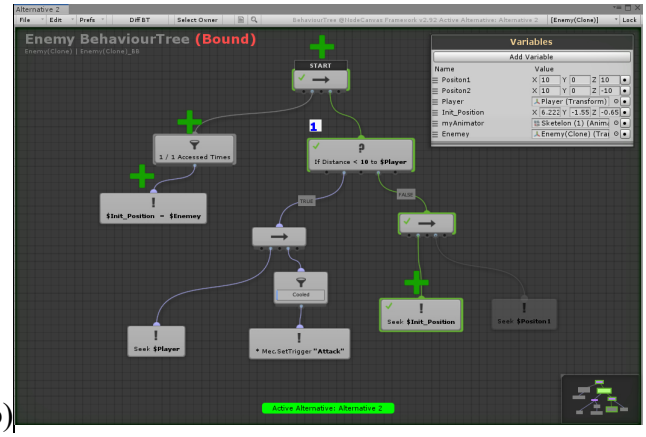


Figure 3-7. BT Differences tab.

(a)



(b)



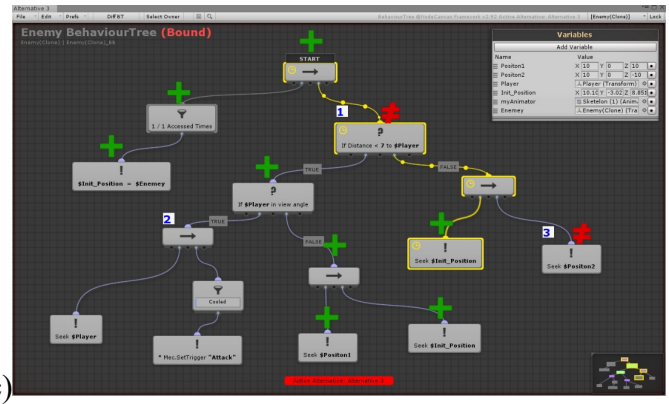
(c) 

Figure 3-8. Difference visualizations in the enemy BT alternatives of the RPG game of Alternative 1 (a), Alternative 2 (b) and Alternative 3 (c).

In *NCAlt* difference visualization of BTs is marked between multiple alternatives against a chosen alternative. The user can choose any alternative as a *reference* to compare against. By default, the chosen reference is the alternative with the BT appearing in the *NCAlt* main tab (Figure 3-1). This is what we refer to as the *one-to-many diffing* technique.

To show the differences between multiple BT alternatives, the nodes are highlighted differently. In the difference visualization mode, differences are visualized for added, deleted, modified nodes, and nodes that changed priority in all of the compared alternatives. To get into this mode, the user clicks on a GUI button called "diff BT" (Figure 3-1g). A diff BT menu tree (Figure 3-7) then appears where the user specifies the types of differences s/he wants to see in the behavior. All the differences mentioned above are visualized in Figure 3-8. A "+" green icon is displayed on top of the node to visualize the added node. Deleted nodes are displayed by reducing the transparency of the node. Since BT executes from left to right, a node on the left has a higher priority. A number is displayed in the top left of the node to show priority change. Lastly, a red "≠" icon is displayed on top of the node to show that the node has been modified.

We will use the previous worked example of John to demonstrate difference visualizations. As shown in Figure 3-1, John created three BT alternatives for an enemy NPC in the RPG game. The white NPC is the active NPC in the game. The green, and red NPCs are the alternative NPCs, each having their alternative BT (Figure 3-8). Alternative 1 was chosen as the active (reference) BT alternative (Figure 3-8a). The nodes SEEK $PLAYER, COOLED, MEC.SET TRIGGER "ATTACK" and second "→" are unchanged between all three alternatives.

Added nodes

In Alternative 2 (Figure 3-8b), "→", 1/1 ACCESS TIMES, INIT_POSITION=ENEMY, SEEK INIT_POSITION nodes are the newly added. In Alternative 3 (Figure 3-8c), two "→", 1/1 ACCESS TIMES, $INIT_POSITION=$ENEMY, IF $PLAYER IN VIEW ANGLE, SEEK $POSITION1 and two SEEK $INIT_POSITION nodes are the newly added. Therefore, all these added nodes have a "+" green icon on top informing the user that they were newly added relative to Alternative 1.

Deleted nodes

In Figure 3-8b, SEEK $POSITION1 node is deleted in Alternative 2 relative to Alternative 1. This is visualized using reduced transparency. The deleted node is also disabled and cannot be executed. This approach of decreasing transparency was found to be effective in the previous work [120].

Modified nodes

In Alternative 3, DISTANCE <10 TO PLAYER was changed to DISTANCE <7 TO PLAYER and SEEK $POSITON1 was changed to SEEK $POSITION2. These changes are all relative to

Alternative 1 because Alternative 1 is selected as the active (reference) BT alternative. Both these nodes have red "≠" icon displayed on top of them.

Priority change

Changing the priority of a node is done either from left to right or right to left, or by changing the parent of the node. In Alternative 1 (Figure 3-8a), IF DISTANCE <10 TO PLAYER was the prime node but in Alternative 2, it became a child of "→". Since the parent node was changed, the priority of this node was also changed and "1" is displayed on top left of this node. "→" is tagged with "2" because its parent was changed from IF DISTANCE <10 TO $PLAYER to IF $PLAYER IN VIEW ANGLE. Since SEEK $POSITION1 in Alternative 1 is modified to SEEK $POSITION2 in Alternative 3, they are the same node with a changed parameter and this node was moved from the left child to the right child. So, "3" was tagged on top left of SEEK $POSITION2.

3.3 Selective Merging of Behavior Trees

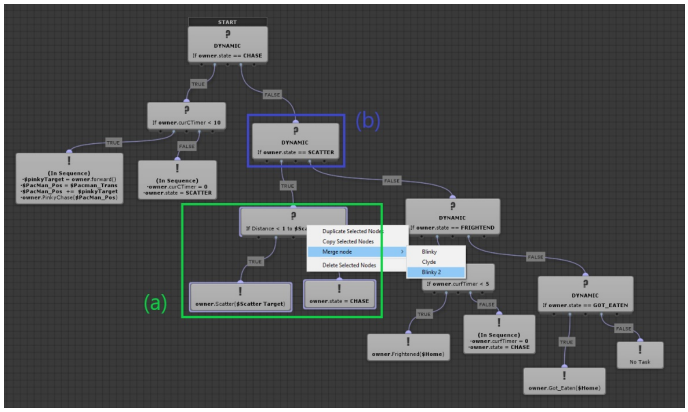


Figure 3-9. Pinky's BT alternative of *Pac-Man.*

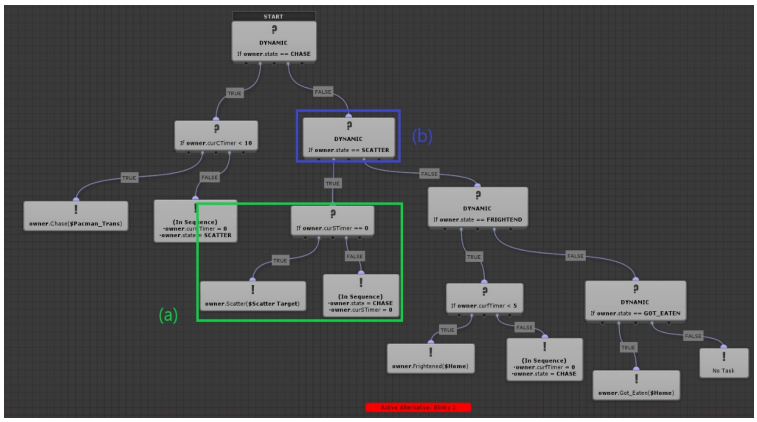


Figure 3-10. Blinky 2's BT alternative before merging with Pinky's BT alternative of *Pac-Man.*

NCAlt supports selective merging of BTs and branching out a node or nodes to explore different BT alternatives. Nodes that do not exist in the target BT are created and connected with the parent node. The nodes are assigned the same ID as in the source alternative. If the parent node is not found, the user is asked for a resolution. If the user decides to merge without a parent, the selected node(s) is/are not connected with any node in the target alternative. If the same node already exists in the target alternative, a new ID is created for that node.

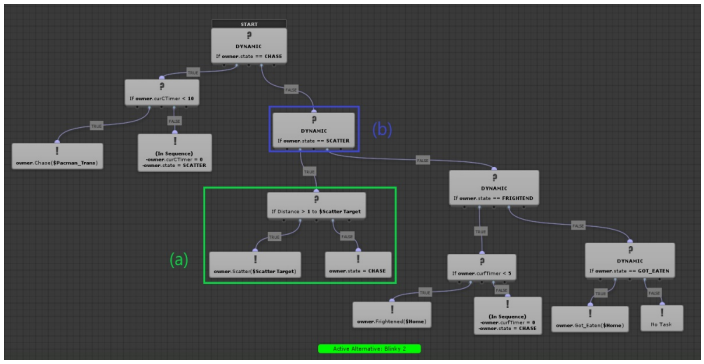


Figure 3-11. Blinky 2's BT alternative after merging with Pinky's BT alternative of *Pac-Man.*

Figure 3-9, Figure 3-10 and Figure 3-11 illustrate a merging scenario in *NCAlt* on a worked example with Sarah. Recall that Sarah created three BT alternatives: Blinky, Clyde, and Pinky (Figure 3-5, Figure 3-6). In the first scatter mode Blinky (Figure 3-5a) will stay in the scatter mode for 10 seconds and then switch to the chase mode. In Pinky's scatter mode (Figure 3-9), he changes his mode from scatter to chase after arriving in the scatter position. Sarah decides to make Blinky's scatter mode similar to that of Pinky's. However, she does not want to get rid of Blinky's current BT. So, she creates another alternative for Blinky's BT and renames it as "Blinky 2" (Figure 3-10). In the BT, the position of nodes is important because the children are executed from left to right.

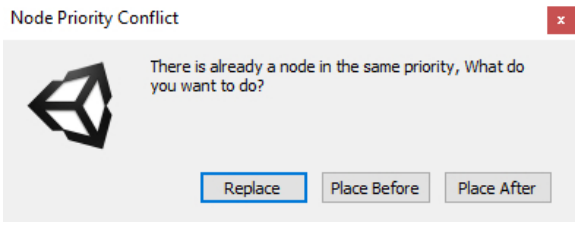


Figure 3-12. A dialog box asks the user to resolve a node merging conflict.

Thus, while merging BT nodes, if there is no conflict, the selected node(s) is/are automatically merged in the target alternatives. However, if there is already a node or nodes in the same position of the tree, a dialog box will appear with the three options as follows: "Replace", "Place Before" and "Place After" (Figure 3-12). The selected node(s) in the target alternatives graph is/are automatically merged depending on the option chosen. In Sarah's case, there was a conflict because there were already three nodes in the same position (Figure 3-10a), so a dialog box appeared (Figure 3-12) asking the user to resolve the node merging conflict. Sarah selects "Replace" because she wants to replace Blinky 2's scatter behavior with Pinky's scatter behavior. Then, the selective merge is completed (Figure 3-11).

Chapter 4
User Study I

We performed a user study to compare *NodeCanvas*, which served as a control, against *NCAlt*. The research goals of the study were to compare the two systems in terms of

1) overall usability and usefulness;

2) effectiveness for authoring multiple BT alternatives, selective merging, difference visualizations;

3) support for exploration;

4) the overall process.

4.1 Participants

We recruited 16 paid participants (three females) from relevant technical undergraduate and graduate programs. The participants' backgrounds were game development and computer science. All participants were between 20 and 29 years old (M = 23.8, SD = 2.6) and had on average 16.5 years of experience using a desktop/laptop computer (SD = 4.3 years). All the participants had between 2 and 7 years of experience with *Unity* (M = 3.4, SD = 1.5), and spend on average 6.5 hours (SD = 3.7) working with *Unity* per week. Six participants had on average 0.7 years (SD = 0.64) of experience with *Unreal Engine*. Two participants had one year of experience with *Lumberyard Engine*. All participants on average had 1.5 (SD = 1) years of experience with at least one node-based programming language or system. Six participants previously worked with *Unreal Engine Blueprints*, three — with *Blender Node Editor*, two — with *Flow Canvas*, two — with *Nodebox,* one — with *CVNodes*. Five participants on average had

0.6 years (*SD* = 0.76) of experience working with block-based system programming languages, such as *Scratch*. One participant previously worked with *Game Maker Language* for four years. We also asked the participants to rate their level of proficiency with general-purpose programming languages, such as C++, C#, and Java. The participants on average had 5.58 years (*SD* = 0.55) of experience with these programming languages.

4.1.1 *Power Analysis*

Based on a previous study that employed a similar design [21], we aimed to investigate and also anticipated a large effect size ($\eta^2_P = 0.14$ or higher). An *a priori* power analysis for a repeated measures F-test with within-between interaction revealed that we would need 16 participants to achieve a recommended minimum statistical power of $1 - \beta = 0.8$, see Field et al. [34], p. 58.

4.2 Apparatus

We used a workstation with 16 GB RAM, an 8-core AMD *Ryzen 7 1700* 3.9GHz, Nvidia *GTX 1060* 3GB with Microsoft Windows 10. A generic IOGEAR keyboard and mouse were used for input. The workstation was connected to a triple-monitor setup. The left monitor was used for pre and post-study questionnaires between the tasks and for displaying all the *NodeCanvas* and *NCAlt* windows for showing the difference visualizations in the BT. The middle monitor was used to display the *Unity* main editor and the gameplay view. The right monitor was used for *NodeCanvas* or *NCAlt* main canvases. All three monitors were recorded using *Open Broadcaster Software* [88], a free and open-source capture software. The investigator used secondary standalone laptop in order to record notes on observations throughout the session.

4.3 Procedure

4.3.1 *Tutorials and Tasks*

We designed two tasks. One task was performed using the standard *NC* (short for *NodeCanvas*) system, and the other was performed using *NCAlt*. Participants were given two *Unity* game project templates that contained all the game objects (PacMan, Clyde, Skeleton, Player RPG, etc.) and the game level scenes for *Pac-Man* and the RPG. They were asked to create an AI for a specific game object. Each of the tasks was preceded by a written tutorial where the corresponding task and template were thoroughly explained. For *NC* the tutorial covered how to create these different types of nodes and connect them and how to write scripts for a node. For *NCAlt* the tutorial also covered how to create and manage alternatives, show difference visualizations and perform selective merging. The participants were asked to perform a warm-up task as shown in the tutorial, so that they are comfortable to use the system when they started the actual tasks.

For the actual tasks, participants were asked to create three BT alternative behaviors. In *NCAlt,* they used the multiple BT alternative option for creating alternatives and, in *NC,* they copied the whole BT on the same canvas. One task involved developing AI behaviors using BTs for three ghosts in a game inspired by *Pac-Man* [79], identical to what we described in the worked example with Sarah. All the ghosts' BT alternatives are shown in Figure 3-5. The movement of Pac-Man was pre-written, so the participants only needed to work on the AIs of those ghost characters (Blinky, Pinky, and Clyde). Some of these variables and functions for these ghosts were written in the script, such as ghost states, move function, etc. The second task involved creating BT alternatives for an enemy NPC of the RPG game, described in the worked

example with John above. The movement of the main player character was already implemented. The participants only needed to work on the AIs of the enemy character (Skeleton). The *Pac-Man*'s ghost and the RPG skeleton enemy both had an empty BT created for to them already. Therefore, the participants had to create the BT from scratch.

4.3.2 *Experimental Design, Independent & Dependent Variables, and Data Collection*

The study used a 2 × 2 mixed factorial design. The independent within-subject variable was System (*NC* or *NCAlt*). The between-subject variable was Order (*NC→NCAlt* or *NCAlt→NC*) in which participants used the systems. We further counterbalanced the study by adjusting both the order of the systems and the order of the tasks (Pac-Man or RPG Game). See Table 4-2**.**

Participant #	**Task 1**	**Task 2**
P1, P5, P9, P13	Pac-Man with *NC*	RPG with *NCAlt*
P2, P6, P10, P14	RPG with *NC*	Pac-Man with *NCAlt*
P3, P7, P11, P15	Pac-Man with *NCAlt*	RPG with *NC*
P4, P8, P12, P16	RPG with *NCAlt*	Pac-Man with *NC*

Table 4-2. Participants' orders, systems and tasks.

The tasks involved limited possibilities for creative expression. Standard tools exist to evaluate creativity support [18], but since this was not an option we focused on evaluating usability and usefulness of *NCAlt* using the System Usability Scale (SUS) questionnaire [12]. A self-developed questionnaire was used to evaluate study specific questions such as effectiveness of authoring multiple BT alternatives, selective merging, difference visualizations, and support for exploration. Participants completed these

questionnaires after completing the task with each system. At the end, a semi-structured interview was conducted to gather more feedback and to have the participants expand on some of their experiences and observations of the systems. This allowed us to capture subjective impressions about overall process, which the questionnaires could not necessarily cover. We also used activity logs (see Longitudinal Study below). By collecting four different types of data we strived to achieve triangulation in user experience evaluation [90]. On average, a full session lasted approximately two hours.

4.4 Quantitative Results

4.4.1 *SUS Scores*

The SUS questionnaire was introduced in 1986 by John Brooke as a "quick and dirty" way to measure the usability of products [12]. Pettersson et al. [90] identified this standardized method as one of the most common for usability evaluation in their recent meta-analysis. Figure 4-14 summarizes the ratings from the SUS questionnaire in a divergent stacked bar chart. The participants rated *NCAlt* equal or higher than *NC* for all the questions. We conducted a two-factor mixed ANOVA test on the SUS score. The main effect of System was significant, $F(1,14) = 40.38$, $p < .0001$, $\eta^2_P = .74$. The SUS score for *NCAlt* ($M = 74.9$, $SD = 7.47$) was higher than for *NC* ($M = 65$, $SD = 9$). The main effect of Order was also significant, $F(1,14) = 14.17$, $p < .01$, $\eta^2_P = .5$. Participants who were exposed to *NC* first ($M = 75$, $SD = 5$) produced higher ratings overall compared to those who were exposed to *NCAlt* first ($M = 65$, $SD = 10.43$). The interaction between System and Order was significant too, $F(1,14) = 9.65$, $p < .01$, $\eta^2_P = .4$. A pairwise t-test with Bonferroni adjustment revealed that participants who were exposed to *NC* first rated *NC* higher ($M = 72.5$, $SD = 2.98$) than the participants who were exposed to *NCAlt* first ($M = 57.68$, $SD = 6.2$), $p < .001$. Participants who were

exposed to *NCAlt* first rated *NC* lower than *NCAlt* ($M = 72.25$, $SD = 8.5$), $p < .01$; they also rated *NC* lower than *NCAlt* than those who were exposed to *NC* first ($M = 77.5$, $SD = 5.5$), $p < .0001$. See Table 4-3 and Figure 4-13.

	NC→NCAlt.NC	*NCAlt→NC.NC*	*NC→NCAlt.NCAlt*
NCAlt→NC.NC	***	-	-
NC→NCAlt.NCAlt	ns	****	-
NCAlt→NC.NCAlt	ns	**	ns

Table 4-3. Results of a pairwise t-test with Bonferroni adjustment for Order × System. Significance cut points: ****: $p < .0001$, ***: $p < .001$, **: $p < .01$, *: $p < .05$, ns > 0.5.

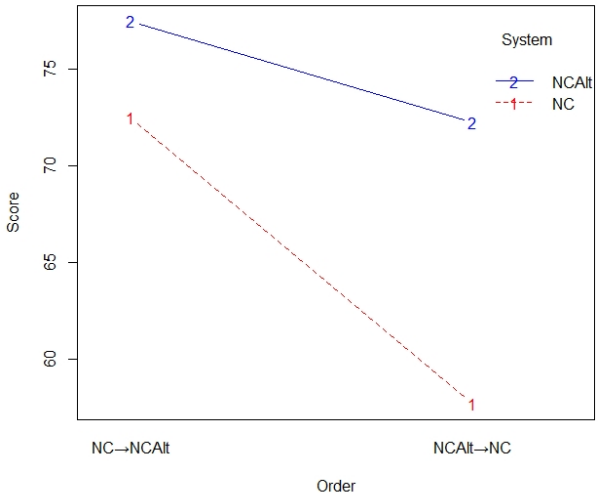


Figure 4-13. Interaction of System and Order for SUS score.

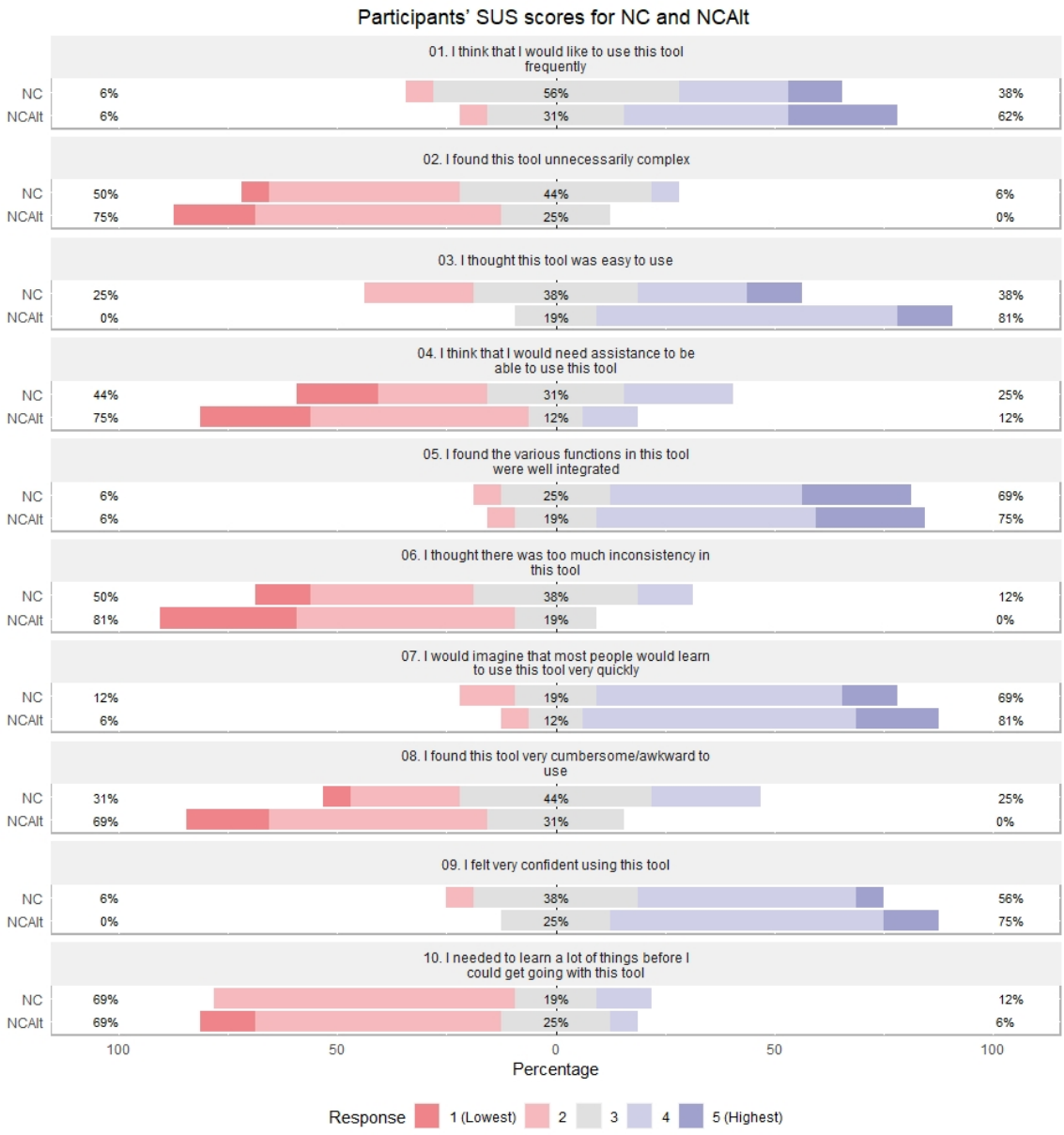


Figure 4-14. Participants' SUS scores for *NCAlt* and *NC* (questions alternate between positive and negative wording).

	Chance of Future use		Efficiency		Ease of use		Multiple BTs		Selective merging		Diffing: game scene		Diffing: BT		Overall	
System	*NC*	*NCAlt*	*NC*	*NCAlt*	*NC*	*NCAlt*	*NC*	*NCAlt*	*NC*	*NCAlt*	*NC*	*NCAlt*	*NC*	*NCAlt*	*NC*	*NCAlt*
Median	4	6	5	5	5	5	2.5	6	3	4	3	6	3	5	4.5	5.5
Effect of Order	$F_{ATS}(1) = 8.56, p < .01$		$F_{ATS}(1) = 3.67, p > .05$		$F_{ATS}(1) = .02, p > .05$		$F_{ATS}(1) = 6.58, p < .05$		$F_{ATS}(1) = 7.87, p < .01$		$F_{ATS}(1) = 3.55, p > .05$		$F_{ATS}(1) = 8.02, p < .01$		$F_{ATS}(1) = 19.1, p < .0001$	
	$Q(1, 9.04) = 8.01, p < .05$		$Q(1, 9.72) = 5.86, p < .05$		$Q(1, 7.93) = 0, p > .05$		$Q(1, 8.74) = 4.14, p > .05$		$Q(1, 8.56) = 7.38, p < .05$		$Q(1, 9.9) = 2.9, p > .05$		$Q(1,10) = 8, p < .05$		$Q(1, 6.8) = 20.28, p < .01$	
Max p.sig.	*		ns		ns		ns		*		ns		*		*	
Effect of System	$F_{ATS}(1) = 11.04, p < .001$		$F_{ATS}(1) = .26, p > .05$		$F_{ATS}(1) = .17, p > .05$		$F_{ATS}(1) = 64.75, p < .0001$		$F_{ATS}(1) = 3.73, p > .05$		$F_{ATS}(1) = 32.53, p < .0001$		$F_{ATS}(1) = 6.72, p < .01$		$F_{ATS}(1) = 5.77, p < .05$	
	$Q(1, 9.91) = 15.5, p < .01$		$Q(1, 8.55) = .22, p > .05$		$Q(1, 9.87) = .12, p > .05$		$Q(1, 7.94) = 55.84, p < .0001$		$Q(1, 9.51) = 3.48, p > .05$		$Q(1, 9.54) = 30.1, p < .001$		$Q(1, 9.93) = 5.29, p < .05$		$Q(1, 8.14) = 10.52, p < .05$	
Max p.sig.	*		ns		ns		****		ns		***		*		*	
Interaction	$F_{ATS}(1) = .52, p > .05$		$F_{ATS}(1) = .22, p > .05$		$F_{ATS}(1) = .4, p > .05$		$F_{ATS}(1) = .62, p > .05$		$F_{ATS}(1) = 2.9, p > .05$		$F_{ATS}(1) = .14, p > .05$		$F_{ATS}(1) = .57, p > .05$		$F_{ATS}(1) = .05, p > .05$	
	$Q(1, 9.91) = 0.19, p > .05$		$Q(1, 8.55) = .22, p > .05$		$Q(1, 9.87) = .49, p > .05$		$Q(1, 7.94) = .55, p > .05$		$Q(1, 9.51) = .32, p > .05$		$Q(1, 9.54) = .37, p > .05$		$Q(1,9.93) = 1.04, p > .05$		$Q(1, 8.14) = 0.56, p > .05$	
Max p.sig.	ns		ns		ns		ns		ns		ns		ns		ns	

Table 4-4. Results of ANOVA-type statistic tests and robust mixed ANOVA based on trimmed means (20%) tests on the participants' ratings of the systems grouped by ranking category. Results that are in disagreement between the tests are shaded in pink. Significance cut points: ns: $p > .05$, *: $p < .05$, **: $p < .01$, ***: $p < .001$, ****: $p < .0001$.

4.4.2 *Self-Developed Questionnaire*

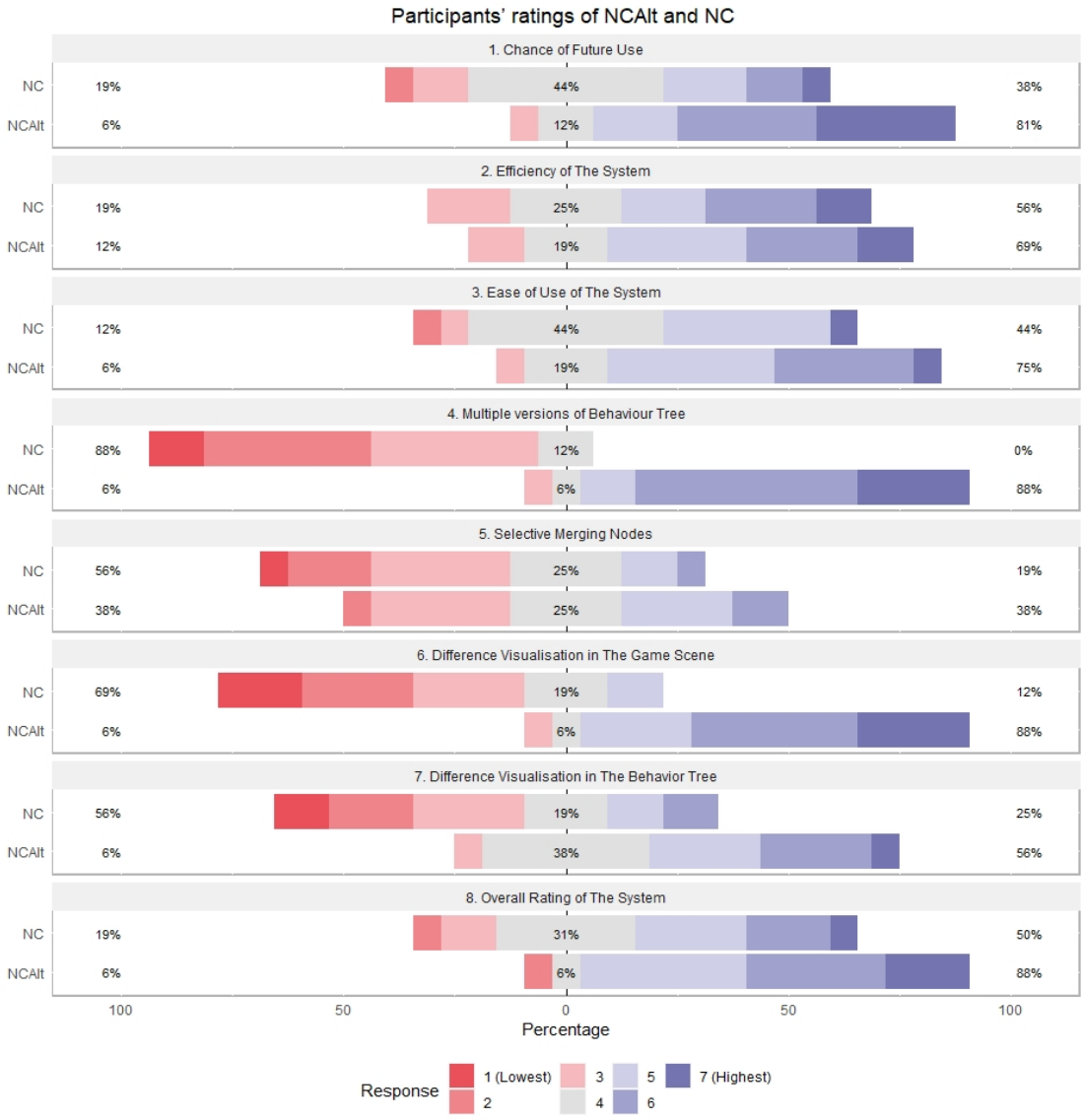


Figure 4-15. Participants' ratings of *NCAlt* and *NC*.

In our self-developed questionnaire, we asked the participants to rate *NC* and *NCAlt* on a Likert scale from 1 (lowest) to 7 (highest) for the chance of future use, efficiency, ease of use, and support for multiple versions of BTs, selective merging, difference visualizations (tree and scene) and overall. A divergent bar chart summarizes the results in Figure 4-15. Table 4-4 summarizes the results of ANOVA-type statistic tests – a non-

parametric alternative to ANOVA for factorial experiments [83] on the participants' ratings of the systems grouped by ranking category. For validation, the results of robust mixed ANOVA tests based on trimmed means (20%), which is another alternative to factorial ANOVA [34] (see p. 643), is also included. We chose to run these tests over Wilcoxon signed-rank tests because the latter is a single factor test. For all the combinations of the categories and factors the two tests gave results that were in agreement, except for the order effects for efficiency and multiple versions of behavior trees. Significance cut off points represent the most conservative of results of the two. The participants rated *NCAlt* higher than *NC* for chance of future use, support of multiple BT alternatives, support for diffing of BT alternatives and diffing in the game scene, and overall. These differences were significant. The remaining differences were not significant (efficiency, ease of use, selective merging). The main effect of Order was not significant for all the categories except for chance of future use, selective merging, difference visualization in BT and overall. The interaction effect was not significant in any of the ranked categories.

4.5 Qualitative Results

We administered a semi-structured interview with the participants at the end of the study. During this time, the participants were asked to express their opinions, to provide feedback on the overall experience, and to explain the reasoning behind their decisions. Overall, the results indicate that participants found the alternatives interface of *NCAlt* intuitive because all the button labels and tooltips clearly identify their works. Participants who had previous experience working in node-based systems prior to the user study felt that *NCAlt* gave them more confidence and they were interested in using it in the future. However, participants without this experience were less interested

because they found both *NC* and *NCAlt* logic a little complex. In the following section, we detail these results in the context of our interview evaluation criteria.

4.5.1 *Ease of Use*

All the participants were easily able to follow our instructions for using *NCAlt*. Participants with prior experience in *Unreal Engine Blueprints* and *Flow Canvas* found both *NC* and *NCAlt* really familiar and felt confident they could learn it independently too. All the participants stated that *NCAlt* was simple for the tasks given. P1 stated, "*I found it really useful to test multiple AI behaviors of a character and being able to switch between the BT alternatives is super easy*". P2 stated, "*NCAlt makes stuff much modular, so the development process is more organized in NCAlt compared to NC*". *NCAlt* had a very simple and organized BT selector panel, which makes it really easy to swap between alternatives. On the other hand, it was difficult for the participants to keep track of all the alternatives in *NC* because they were all sharing the same canvas. P10 stated, "The *learning curve from NC to NCAlt is almost nothing*". Our findings in Figure 4-13 also agree with this statement. The participants who were introduced to *NC* first and then *NCAlt* felt comfortable with all these features of *NCAlt*.

4.5.2 *Exploration*

P2, P5, P9, P15 mentioned that they were more willing to try out different things using *NCAlt* because it is less of an inconvenience compared to *NC*. All the participants stated that *NCAlt* helps them to explore more ideas. P8 stated, "*I won't be afraid to take more risk. I would just create a new alternative and if there is a mistake, I can easily move back to the previous one*". Even though in our study the participants did not have much chance to show their creativity P8 still stated: "*I think exploring more ways will be making my game AI better*". Both P6 and P7 mentioned that canvas gets really messy

with the *NC* and, as a result, they will not consider creating alternatives in *NC* in the future. P4 stated, *"I am more inclined to make small changes and try out them easily in NCAlt compared to NC*".

4.5.3 *Speed*

Originally this criterium was called *process* but the collected responses emerged to mostly reflect speed. P2, P4, P9, and P12 thought that *NCAlt* makes the process of developing AI much faster. To support that P9 said, *"NCAlt helps iterative testing much faster because I can try a bunch of ideas very fast"*. All the participants really liked the feature of *NCAlt* which allows opening multiple BTs at once. This helped them to manipulate the trees in parallel and compare the AIs in a side-by-side simple workspace. To support that, P8 stated that *"Using NCAlt, I can easily see exactly at which point the AI changes its behavior"*. This suggests that *NCAlt* can help game developers in debugging their game AI. P2, P3, P6, and P9 also mentioned during the interview that *NCAlt* helped them to debug much faster. P5 and P11 think that *NCAlt*'s selective merging makes the process of editing alternatives a lot faster. To support that P11 stated, *"In NCAlt merging is very fast by just right-clicking and selecting the alternative*". However, some of the participants were still not able to use it properly. They were mixing it with the typical copy and paste operation. The reason behind this could be that selective merging is not very common and is a very new method.

4.6 Criticism, Suggestions & Uncovered Issues

We received some useful feedback on how to improve the system further. P4 stated *"While merging, it took a couple of seconds to find where it merges"*. Although merging nodes were marked lightly, they were still hard to see for P4. So, P4 suggested making the area where the selective merge happened more visible. We modified *NCAlt*

to highlight those merged nodes with higher brightness.

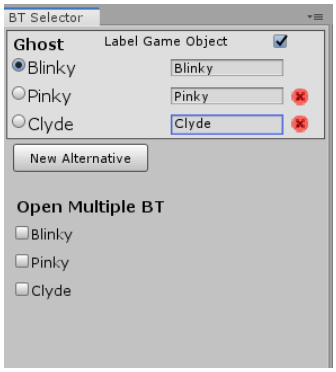


Figure 4-16. An example of BT selection tab after adding the delete buttons and checkboxes for opening multiple BTs.

While creating alternatives, P10 mistakenly created a new alternative from Alternative 1 in place of Alternative 2. Later on, P10 tried to delete that alternative but there was no option for deleting. So, P10 renamed that alternative "deleted alt". In response, we have added this feature in *NCAlt* so that the users could delete any alternative before the longitudinal study. As shown in Figure 4-16, a small GUI button with the red "X" icon has been added to perform this action. Furthermore, comparing multiple alternatives in the game scene, multiple instances of the same game object were interacting with each other, which hampered the workflow of many. In response, we modified *NCAlt* so that game objects of the compared alternative behave like a ghost game object. The instance created from the same game object does not collide with other alternatives created from the same instance and it can pass through each other as well. To perform this function, we have created a new layer while instantiating alternatives. Layer-based collision detection forces a game object not to collide with other game objects that are set up to a specific layer.

Another issue that we encountered was while opening a BT alternative on a new canvas. Since this panel was available in the inspector panel of *Unity*, the participants felt they were at an inconvenience due to the need to move to the inspector panel when

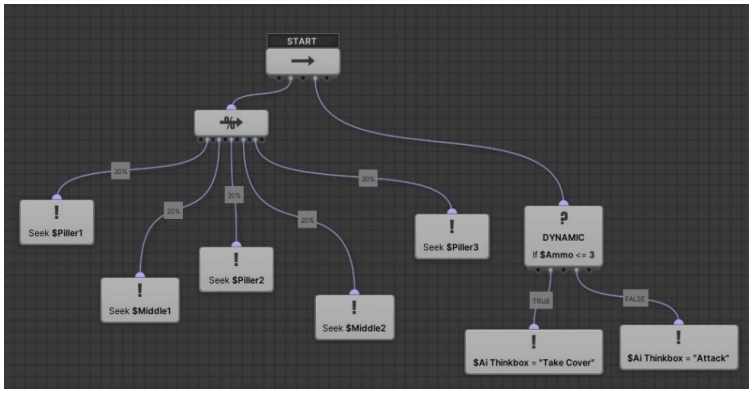
opening multiple alternatives at the same time. Some of the participants also suggested that this panel can be placed inside of the BT Selection panel. Therefore, we have moved those checkboxes inside the BT Selection Panel as shown in Figure 4-16.

Chapter 5

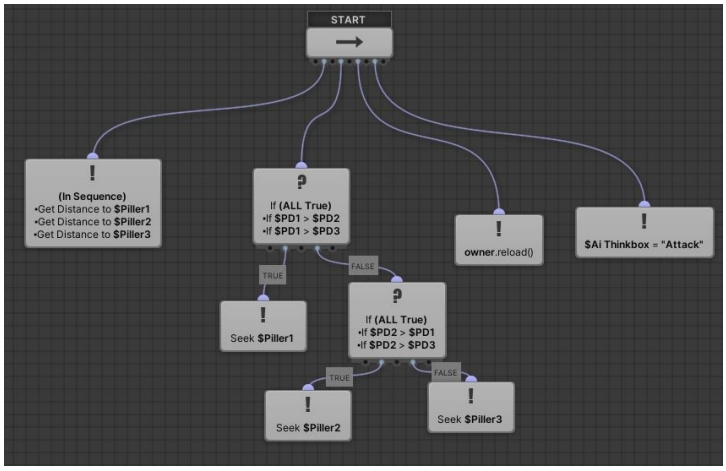
Longitudinal Study

We ran a longitudinal study over a course of two weeks to investigate how the user utilizes alternatives when more freedom is given with an open-ended task.

(a)



(b)



(c)

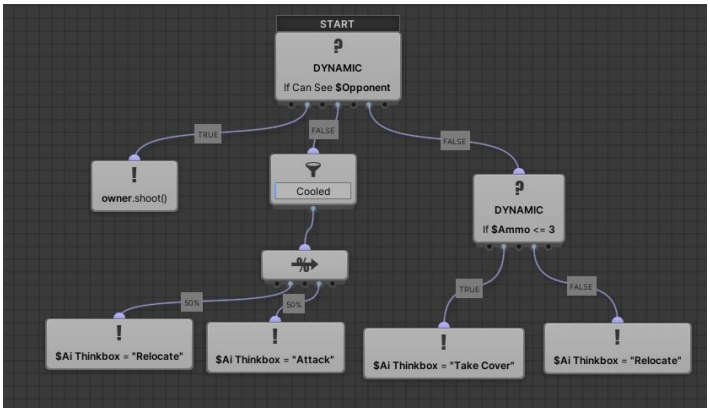


Figure 5-17. Relocate (a), Take Cover (b) and Attack (c) BT alternatives in the two-person shooting game.

We selected one participant from those who completed our first study. In this study, the participant had the freedom to work on any game they desired.

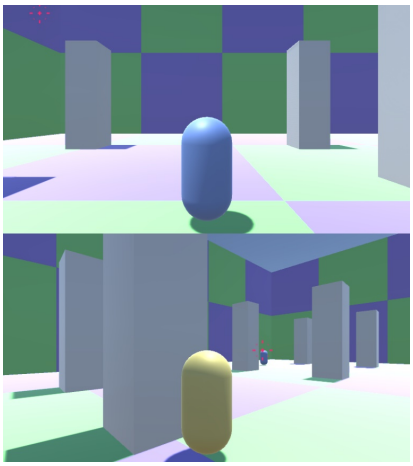


Figure 5-18. A two-person shooting game from the longitudinal user study.

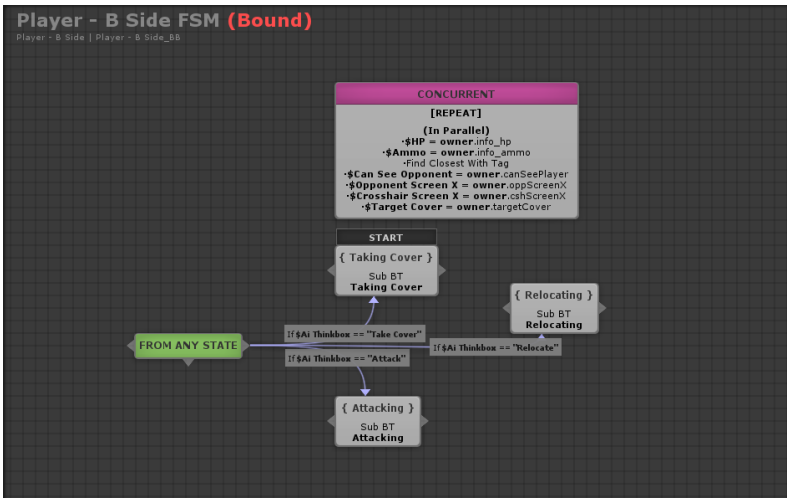


Figure 5-19. An FSM Tree of the two-person shooting game.

The only condition was to have AI present for a game character. We used an activity log to record all the events sequentially such as creating and deleting alternatives, creating and deleting nodes, selective merging and difference visualizations. The participant worked on a two-person shooting game (Figure 5-18) where one character is controlled by the player and another — by an AI. The AI character had three active BT alternatives for three states which were “Relocate”, “Take

Cover” and “Attack” (Figure 5-17). The participant created five BT alternatives (two for Attack, two for Take Cover, one for Relocate) and deleted two alternatives.

In our first study, only one BT alternative was active at a time for a single game object. However, in this study, the participant used a Finite State Machine (FSM) [7] in *NCAlt* (Figure 5-19) where the users define conditions for translating between states and, in our case, translating between BT alternatives. The FSM continues executing its current BT alternative until conditions are satisfied for the transition to another BT alternative.

The participant first created the “Relocate” BT alternative (Figure 5-17a), which basically moved the AI character behind one pillar to another pillar or between the pillars randomly. It would automatically change its state to “Take Cover” or “Attack” based on the amount of ammunition. If ammunition is less than or equal to 3, the AI state changed to “Take Cover”, otherwise it changed to “Attack”.

The participant then created another BT alternative and renamed it as “Take Cover”. The AI reloaded its gun in this state. In the first “Take Cover” BT alternative, the participant used a controller key and timer function to check if the AI character reached the cover position or not. However, the participant did not like how the AI was moving. Thus, the participant created another alternative (Figure 5-17b), where they simply added a SEEK VECTOR3 node to move the AI character to take cover behind the nearest pillar.

After that, the participant created the “Attack” BT alternative where the AI character checked if the main character was visible or not. If it could see the player character, it shot at him/her, otherwise it would change its state to “Relocate” (or if ammunition is less than 3 — to “Take Cover"). Then the participant created another

"Attack" BT alternative (Figure 5-17c) and added a conditional probability node that could choose the state to relocate or attack after every five seconds.

As we can see in the above-mentioned example the participant used multiple BT alternatives while making their own game. According to our activity log, the participant created five BT alternatives and deleted two BT alternatives. We found in our activity log that the participant used *NCAlt* for nine days over a period of two weeks and on average the participant used *NCAlt* approximately 68 minutes each day. In addition, we also found out that the participant worked on two BT alternatives (Relocate and Take Cover) during the first four days, and created on average eight nodes on each day. However, after four days, the participant was creating on an average 30 nodes per day. By this time, the participant has created five BT alternatives. This may suggest that *NCAlt* became more useful as substantial progress on the project was made, although we cannot discard that this could also be due to the fact that the participant became more familiar with the tool. The participant also used the diff game scene, diff BT and selective merging features 12, 10, and 5 times respectively. This indicates that the participant was willing to use all the newly added *NCAlt* functionality when they were working on a free and open-ended task. Besides, the combination of FSM and BT alternatives enhanced the usability of *NCAlt*. To support that, the participant mentioned during the interview that "*the combination of FSM and BT alternatives make creating AI in NCAlt a lot easier*". The participant suggested to add the support for multiple alternatives for FSM too.

Chapter 6
Discussion of User Study I and Longitudinal Study

Here we interpret and describe the significance of our findings in the light of *NCAlt*.

Firstly, to study the usability of *NCAlt,* we used the SUS questionnaire, where on average in 8 out of 10 questions, the participants rated *NCAlt* equally or higher than *NC* and. For inconsistency in the tool and learning curve *NCAlt* was rated marginally lower. This is not surprising because it was the first evaluation of *NCAlt*, where we learned that there were minor issues with the UI. Also, the learning curve for *NCAlt* is likely a little steeper due to the presence of additional features compared to *NodeCanvas*. However, overall, the results suggest that for the most part *NCAlt* seems to support usability well.

We employed a mixed experimental design where the system order was different for the two groups of participants to counterbalance the potential learning effect and bias of participants always using the systems in the same order. We found that there was a significant interaction between the system and the order for the SUS score, which further revealed that participants who were exposed to *NC* first rated *NC* higher than the participants who were exposed to *NCAlt* first. This shows that the choice to counterbalance the study was the right one and future studies that employ similar evaluations should employ mixed design to deal with potentially significant bias in the ratings. Nonetheless, *NCAlt* was still rated significantly higher on the SUS overall.

The SUS results were in agreement with the participants' ratings for the chance of future use, support of multiple BT alternatives, support for diffing of BT alternatives and diffing in the games scene, and overall, in our self-developed questionnaire (Figure 4-15). However, in the remaining categories (efficiency, ease of use, and selective merging) the differences between *NC* and *NCAlt* were not significant. This was further

corroborated from the qualitative findings as some participants were not sure about the purpose of the feature. It is understandable because this method is new to novice users. We speculate that since all the participants were familiar with at least one node-based programming language, they found both *NC* and *NCAlt* efficient and were able to use them with an ease. Overall, our findings indicate that for the most part the features we introduced have the potential to improve the workflow of the game developers creating game AI using BTs. All the participants liked having access to multiple BT alternatives in *NCAlt* and they desired to have this feature in the original *NC* too. In the final question of our self-developed questionnaire, we asked participants about their preferred choice of the system for the tasks given between *NCAlt*, *NodeCanvas*, or other. All participants preferred *NCAlt*.

During the interview, participants were asked which difference visualization mode they found more useful. All the participants who worked on *Pac-Man* using *NCAlt* really liked the difference visualizations in the game scene compared to difference visualizations in the tree. One possible explanation could be that when the whole game scene is visible to the users, differences in the game scene are useful because the user sees exactly how multiple AI alternatives are different from each other. On the other hand, for difference visualizations in the tree mode, the number of nodes in the Blinky, Pinky and Clyde's tree was 15. The participants needed to pan and zoom multiple times and could not see the entire BT from a readable zoom at once. This suggests that having a larger BT makes it harder to compare multiple BTs. Besides, 7 out of 8 participants who used *NCAlt* for the RPG game preferred difference visualizations in BTs more than in the game scene. One possible explanation could be that in the RPG game, the number of nodes in the first, second and third BT alternatives were respectively 7, 10 and 15. Therefore, it was easier to see the highlighted

differences in the BT. Besides in the RPG game participants often could not see all the enemy BT alternatives in the game scene at the same time because the game scene was too large to be able to fit into the whole game view at once.

Comparing the speed of task completion was not part of our research goals, as we hypothesized the complex nature of the tasks would generate a great amount of variability between participants. Previous user studies on alternatives in similar settings also did not report speed (e.g., [71, 121]). In the future, more focused studies can be made to look at the potential effect of speed, since, interestingly, a number of participants felt like they were faster with *NCAlt*.

Finally, *NCAlt* can be considered promising in the light of game development or any node-based systems. Thus, it is possible that the functionality that *NCAlt* provides for supporting alternatives in BTs can also be desirable in visual programming in general as there is evidence that similar functionality can work for, e.g., *Unreal Engine Blueprints* [71] generative design [121, 120] and text programming [32] as well.

6.1 Limitations

There are notable limitations to the study design and findings. Our goal was to adapt the interfaces that already work in node-based and parametric design tools to game AI development and to evaluate their usability and usefulness, among other things. Although these interfaces have been well received in the communities they were initially created for (media artists and architects), we did not conduct any surveys of professional game developers to determine if the need for these interfaces also exists in that community, neither did we employ professional game developers as participants. Our participants were knowledgeable in game development with experience delivering full games as part of their study, but arguably, they cannot be considered professionals. Therefore, the results of our study are valid with the assumption that the need for the

interfaces for alternatives and difference visualization for behavior trees can exist. A more certain conclusion would be possible, if professional game developers were used as participants. To the best of our knowledge a tool competitive to *NCAlt* does not currently exist. As a result, we evaluated it against *NodeCanvas*, which was not built for the same purpose. Although we counterbalanced the study, it is important to note that the participants have been compensated, which could still generate bias in the study results. Furthermore, we employ convenience sampling and as a result, we are not comfortable to make any broad generalizations [98].

Chapter 7
NodeCanvas Collaboration

7.1 Pre-Development Requirements Gathering

Before development on *NCCollab* started, we wanted to solicit data for our requirements from potential future users. We posted a list of questions online on Unity Forum [110], GameDev Forum [37], Dream.in.code [93], TIGForums [107] and NodeCanvas Discord channel [26]. The list consisted of the following questions:

1. Which tool or a resource do you use while developing games collaboratively?
2. Do you think that using collaboration features in a visual scripting tool (such as *Unreal Blueprints*, *Playmaker*, *Bolt*, *NodeCanvas*) for game development would benefit you during game development? If so, could you please explain how?
3. Tracking history allows access to previous states quickly. Do you think having this feature while developing games collaboratively using a visual scripting tool would be useful? Why? If so, then how?
4. Do you think the ability to perform collaborative game development in real-time using a visual scripting tool would be useful (e.g., real-time collaborative editing in Google *Docs*)? If so, then how? Or do you think asynchronous collaborative visual scripting would be more useful? Why?

7.2 Results of Pre-Development Requirements Gathering

In total we managed to solicit data from a total of eight respondents online. The findings were as follows:

7.2.1 *Question 1*

All the solicited respondents stated that for collaborative development they used *Git* even. This included also those who had experience with only scripting languages.

7.2.2 *Question 2*

One respondent stated: "*Providing an easy-to-understand form of scripting for 2+ users to work within real-time is already a step up from what Unreal Blueprint system provides (no merging, local only). Such a tool would make working with a non-programmer much easier and debugging projects as a group becomes much more streamlined*". Another respondent stated: "*collaboration will be helpful in scenarios where I am working with another team member – such as an artist or designer – who is not as well-versed in the game's code. In cases like this, being able to edit the script together can be helpful. With everything being online right now, collaborative tools can be very helpful to communicate across development teams*". Two other respondents mentioned that it will be useful for prototyping ideas with other people, instead of being stuck in an *echo chamber* where they are on their own and unsure of what decisions to make.

7.2.3 *Question 3*

One respondent stated: "*Tracking history would be extremely helpful for bug checking for example, if you have a previous working version you can easily backtrack to that working version*". Besides, three respondents expressed that they were confused an alternative history tracking mechanism is needed, as according to them, *Git*'s history already does the work. We tried to address this by designing BT history in such a way so that it keeps track of all the changes of the BT while *Git*'s history tracks changes for every push to the server. Also, BT history keeps track of changes automatically without

interfering with the development flow, while with *Git*'s history explicit pushes need to be initiated by the user who decides which significant events need to go on record. While this offers more customization of history, *Git*'s process is arguably more interrupting to the development flow.

7.2.4 *Question 4*

Seven of eight respondents replied to this from a point of view of text-based programming. One respondent stated they always prefer the asynchronous mode for text-based programming but would like to try the synchronous mode for visual scripting. Another respondent stated that "*Synchronous mode would be nice for fast prototyping because two participants can put some concrete idea really fast*". Yet another respondent stated that it would be beneficial to make sure there are no compilation errors before syncing with other users – something *Git*, being language and syntax neutral, does not inherently support. This feature was implemented in *NCCollab*.

7.2.5 *Summary and Implications for Interaction Design in NCCollab*

The responses we received helped us to expand our initial thoughts on a collaborative visual environment for behavioral NPC animations.

From the responses to the first question, we identified that *Git* is used a resource by the programmers use for collaboration. Although this is a common knowledge, while developing *NCCollab*, we tried not to significantly deviate from how *Git* is used in collaboration.

From the responses to the first question 2, we identified use case scenarios for *NCCollab*. Among these is a situation where an artist or designer who are not proficient in programming could edit the scripts together.

From the responses to the first question 3, we identified that BT history which allows access to previous states of BTs is different from traditional *Git* history. This is because history tracking in *NCCollab* is more non-intrusive compared to *Git*, since in *Git* users need to push the code to the server to keep track of history. In contrast, *NCCollab* does this in the background without interfering with the flow of BT development.

From the responses to the first question 4, we identified that both synchronous and asynchronous collaboration could be useful. We also identified that the preference will vary between different users and use cases. As a result, both modes were implemented in *NCCollab* and later evaluated in a user study described below.

7.3 Introduction of NodeCanvas Collaboration

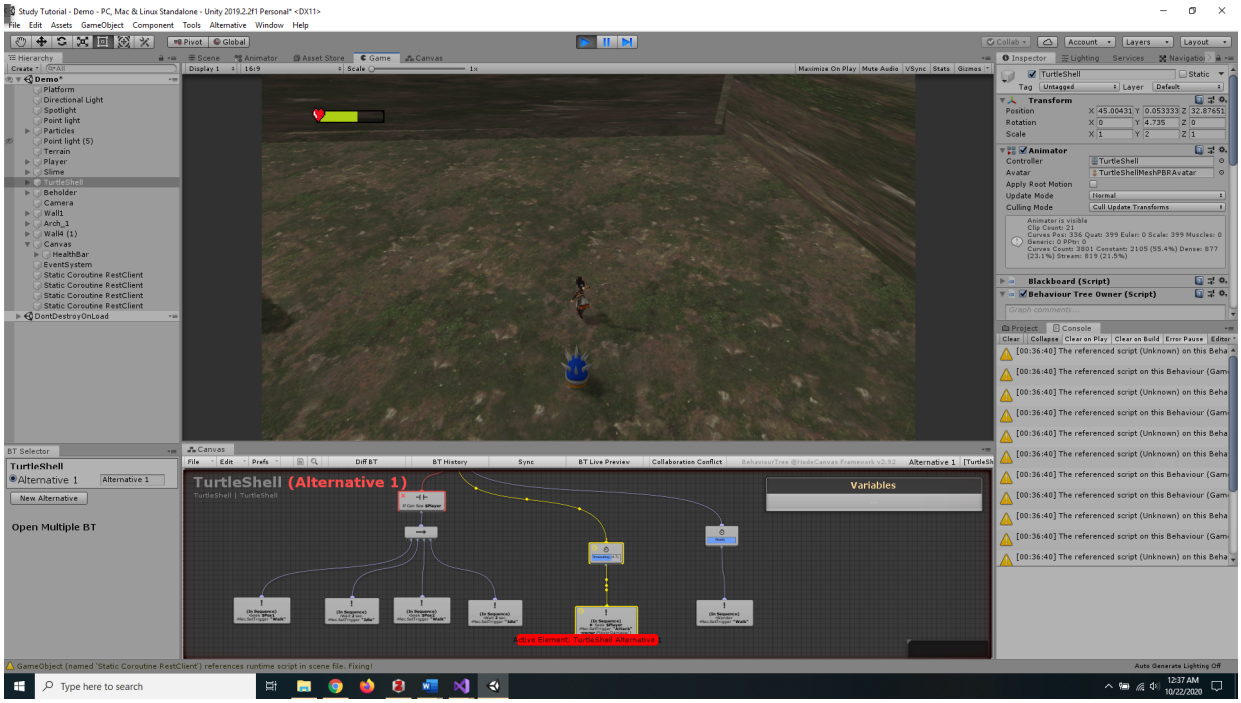


Figure 7-1 *TurtleShell* chasing the player character in the game view.

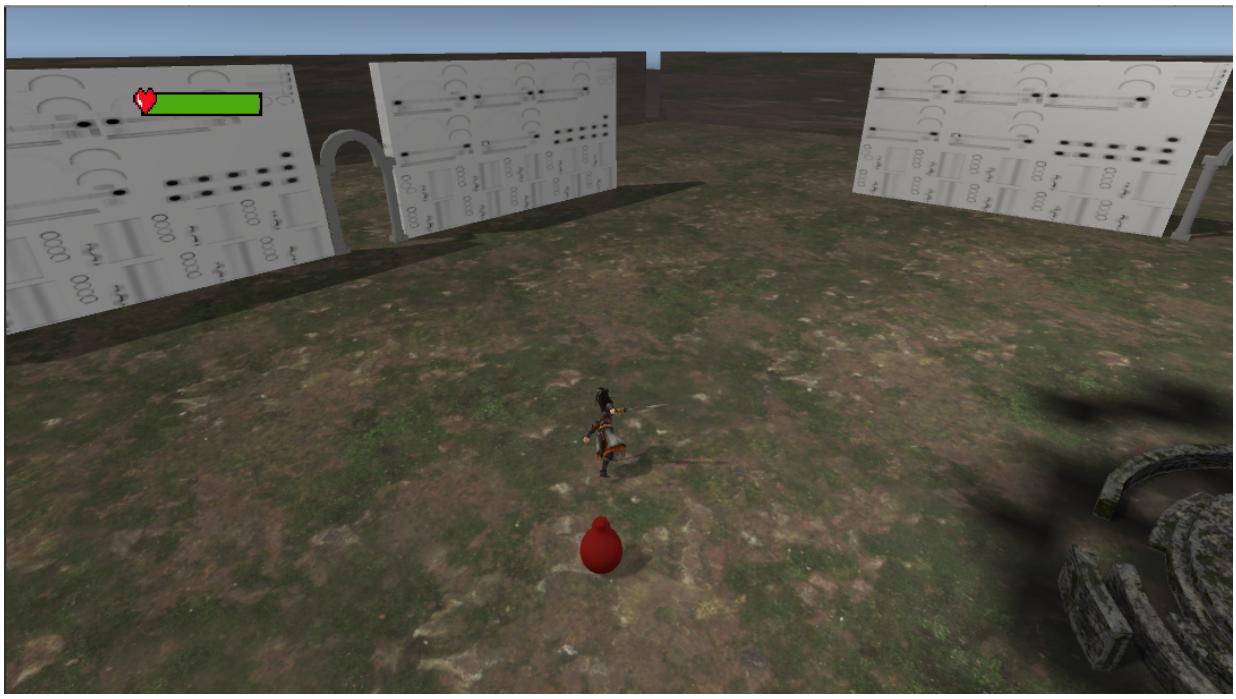


Figure 7-2 *Slime* chasing the player character in the game view.

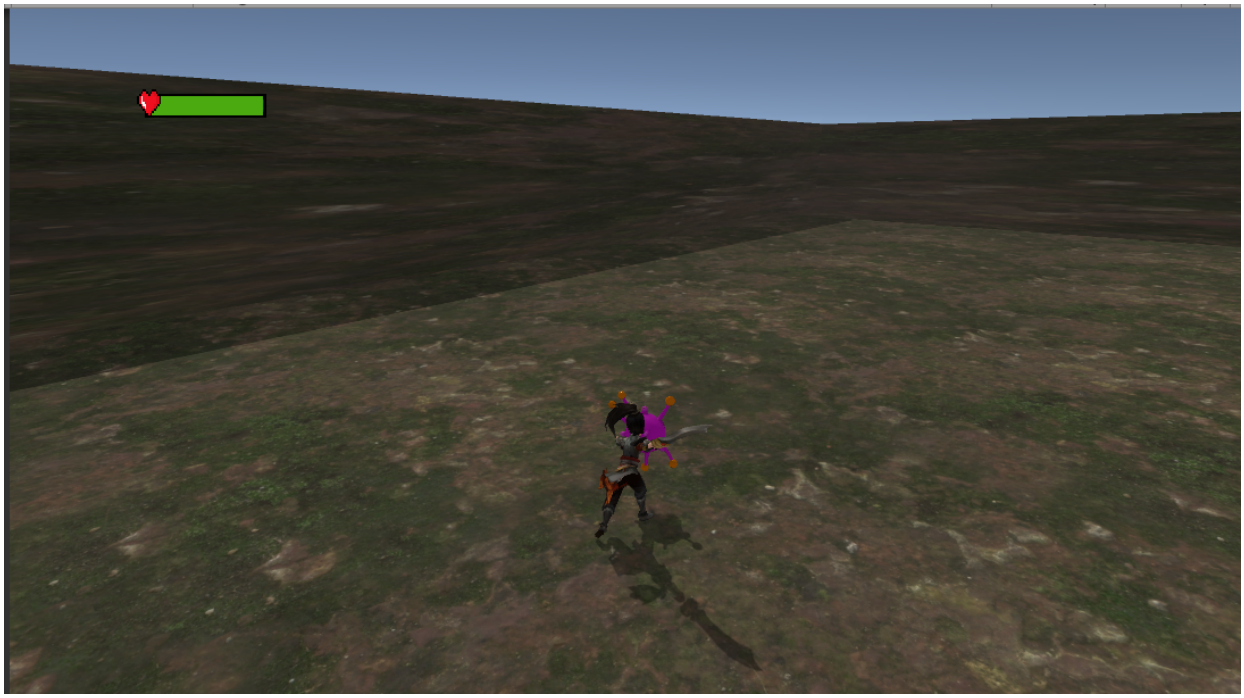


Figure 7-3 *Beholder* being attacked by the player character in the game view.

Here we present *NCCollab* (*NodeCanvas Collaboration*) – an extension to *NodeCanvas* that supports collaboration between game developers so that they can improve the adaptation of unplanned challenges and help co-developers when they are stuck. In

NCCollab, developers can collaborate both synchronously and asynchronously while working on BTs. In order to add collaboration features to a visual programming environment, we need to address two related questions:

1. which concepts needs to be conveyed in these channels,
2. which UIs to use to implement these features?

To answer the first question, we have drawn on several concepts from the previous work and applied them to visual programming. As s result, the following forms of interactions were implemented:

a) two types of collaboration (synchronous and asynchronous);
b) a live preview of other collaborators' BTs and a build-in text messaging option to chat with other collaborators;
c) ability to track and restore of BTs from history;
d) conflict resolution.

To answer the second question, we strived to keep the balance between delivering a variety of functionalities and not overwhelming the user with complex interaction, since this would defeat the purpose of using visual programming in the first place. To accomplish this, we aligned implementing all the auxiliary features with collaboration, which was the focus in the development of *NCCollab*. *NCCollab* was developed on top of *NodeCanvas* by modifying the source code of *NodeCanvas* [82]. *NCCollab* was also integrated to work together with *NCAlt (Node Canvas Alternatives)*, so all the features for creating and managing alternatives in *NCAlt* were made available for use with *NCCollab*. There is no theoretical limit on the number of collaborators that can use *NCCollab* together. On the server side, *NCCollab* uses *Google Firebase* [35]. As a result, there could be a practical limit on the number of collaborators working at the same time due to the limit on the number of responses to the API requests at a time

but we do not foresee this to happen often with normal use. Moreover, *NCCollab* makes sure that it does not sync with the other user's BT if there is a error in the behavior trees or game source code.

7.4 Collaboration using Vanilla NodeCanvas

In the vanilla version of *NodeCanvas,* the users can use *GitHub* [13] or other similar platforms to share the BT's JSON (JavaScript Object Notation) file and collaborate offline. This prevents the users from making independent changes confidently for the fear of making conflicted updates or merging with someone else's tree which may contain errors. To deal with this problem, before development begins, the users must engage in careful planning and coordination about which parts of the BT which users are permitted to modify. Moreover, the users may occasionally need to organize meetings whether virtual or physical to discuss the revision of the BT whenever significant changes were made to make sure that all the users are on the same page and then make further planning. Identifying what constitutes a significant change may not be easy, since each user may have different ideas about what a significant change is, so disagreements may arise about the changes made or scheduling these meetings in first place. This cycle continues until the BT has been fully developed. It is evident, that this process is far from being efficient and that it also can significantly interrupt the flow of development. The asynchronous and synchronous modes were designed to address these issues.

7.5 Asynchronous Collaboration

In *NCCollab*, the user can collaborate with others asynchronously. In this mode, the users work on their own BT. At any time, if desired, they can *sync* with another collaborator's BT. This is done by clicking on the "Sync" button in the *NCCollab*

toolbar (Figure 7-8). The user has the option to work on the same BT, a BT alternative of the same game object, or a different game object altogether.

When our version of *NodeCanvas* is launched *NCCollab* is inactive by default. To enable asynchronous collaboration, the user must check the "Asynchronous collaboration Mode" under the "Prefs" tab as shown in Figure 7-4.

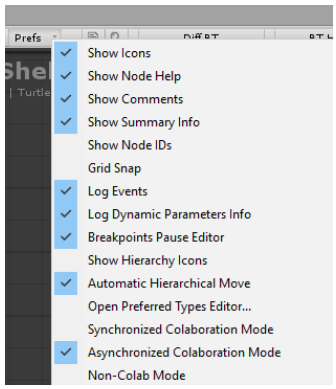


Figure 7-4.Enabling Asynchronous Collaboration Mode in the Prefs tab menu.

When the user enables asynchronous collaboration, *NCCollab* starts sending all the recent BT data (listed below) to the Google *Firebase* real-time database. The data is saved as a JSON file in the Google *Firebase*. To simplify storage in the database and minimize the number of API requests, three pieces of data are transmitted to the server, which are:

- the current BT JSON file that is hosted in the canvas allowing other users to sync with their BTs with;
- a JSON file that consists of a unique ID (*AltId*) for all the nodes that have been created by the users to keep track of which nodes were created by which users, and
- a Boolean variable that sends an update to other users that there has been a change made on this BT or its alternative (if there are multiple of them) by a specific user.

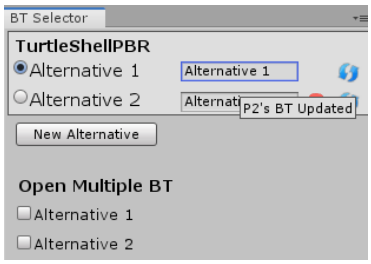


Figure 7-5 BT Selector Window with Update Icon.

A fundamental requirement to support collaboration is awareness. Dourish and Belloti define awareness as "an understanding of the activities of others that provides a context for your own activity" [31]. To support the requirement for awareness in a collaborative system, it needs to provide information about development activities or to notify developers of events relevant to them, such as tree changes, comments, etc. To notify about the changes made in a BT, *NCCollab* uses a refresh icon appearing to the right of the alternative's name in the BT selector window (Figure 7-5). When the users hover their mouse above the alternative's name or the update icon, they can see who have updated the BT as the name of the user appears in the tooltip.

Imagine, two game developers John and Sarah, are trying to create an AI behavior for a game object named *TurtleShell* in a game jam working from home. Figure 7-1 shows *TurtleShell* in the game scene chasing the player. They have an overall idea that their AI character will have three states *patrolling*, *attacking,* and *fleeing*. They decided that John will be working on the patrolling state and Sarah will be working the attacking and fleeing states.

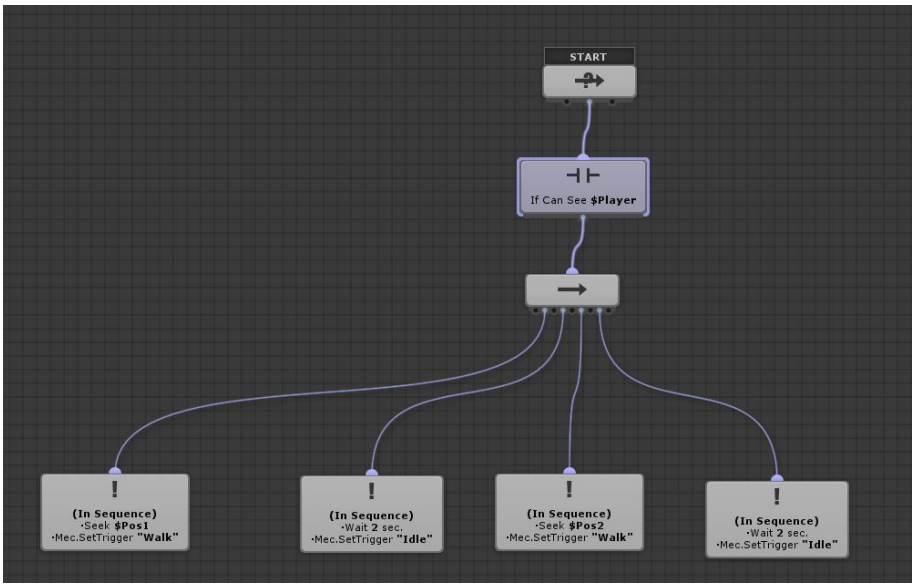


Figure 7-6. John BT of the *TurtleShell* AI Character.

John starts working on the patrolling state. In his vision, the AI character moves from one position to another and after arriving at each position the character rests for 2 seconds. To perform these actions, John creates a selector node, a binary selector node (If CAN SEE $PLAYER), a sequencer node, and four action nodes. John then assigns necessary tasks to the nodes as shown in Figure 7-6. *NCCollab* sends the BT data to the cloud every time an update has been made by John except for if there is an error in the game. Since Sarah is also using *NCCollab* in the asynchronous mode, these changes will not be automatically applied to her BT. However, she can see in the BT selector panel that an update has been made by John and she can also check the live preview of John's BT in another canvas, which is discussed below.

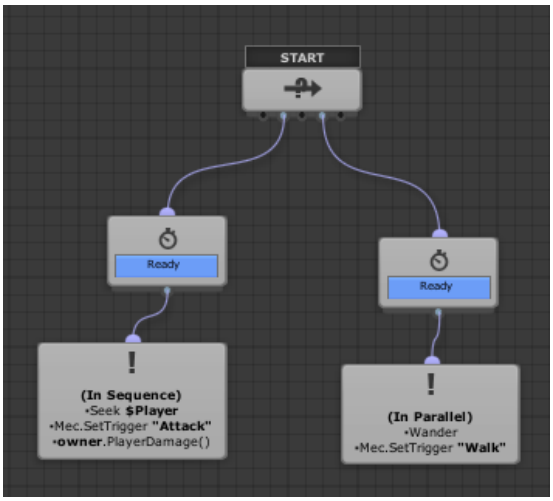


Figure 7-7. Sarah's BT of the *TurtleShell* AI Character.

At the same time, Sarah is also working on the attacking and fleeing states in the *TurtleShell* (AI character) behavior tree. In the attacking state, the AI character starts chasing the player. If the AI character reaches the player, the AI character engages in an attack for 15 seconds. After that, the AI character changes its state to fleeing. In the fleeing state, the AI character wanders around for 15 seconds. Then the AI character changes its state to patrolling. To perform these actions, Sarah creates a selector node, two timeout nodes, two action nodes, and assigns necessary tasks to the nodes as shown in Figure 7-7. After finishing the work, Sarah decides to sync her BT with John's. So, she clicks on the Sync button from the canvas toolbar (Figure 7-8). Now, *NCCollab* automatically syncs Sarah's BT with John's. The newly added nodes after syncing are highlighted using a purple outline as shown in Figure 7-8. Highlighting allows the user to identify the changes after syncing.

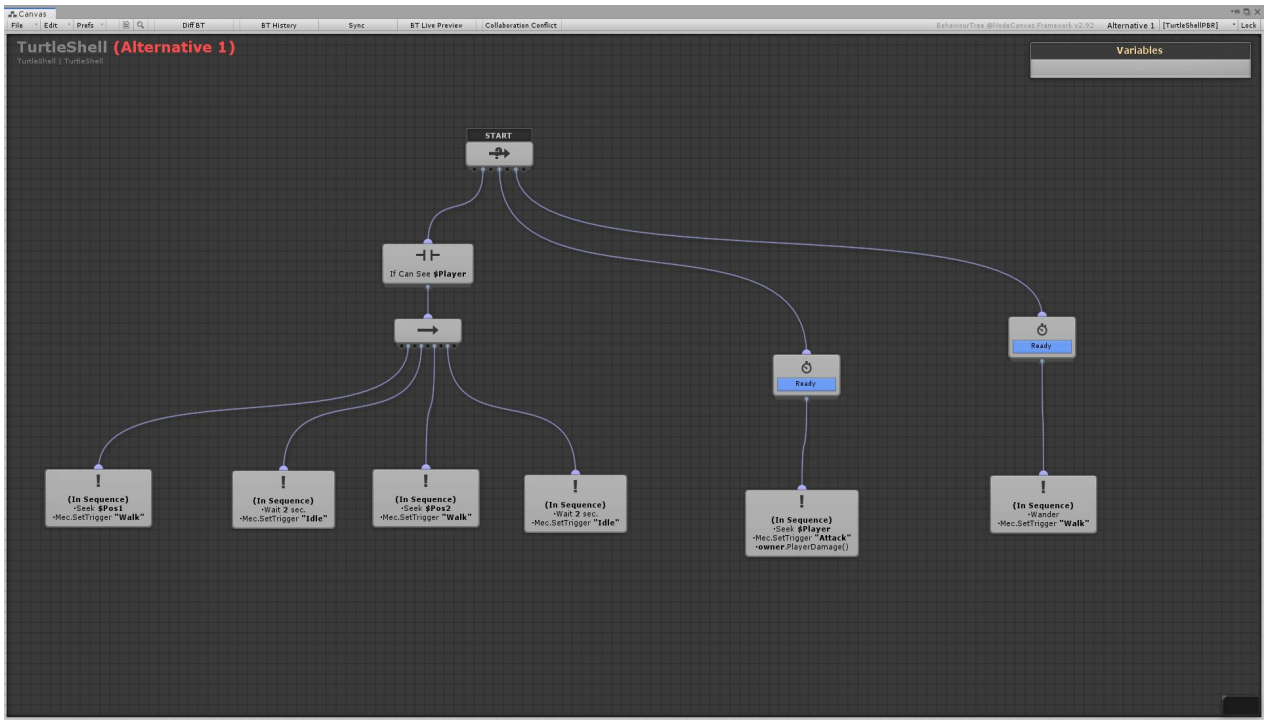


Figure 7-8. Sarah's *TurtleShell* BT after syncing with John's BT.

7.6 Synchronous Collaboration

In *NCCollab*, the users can also collaborate with other users synchronously. The mode is called *synchronous* due to the fact that multiple users can work on the same BT at the same time. In synchronous collaboration, each user has the whole working tree locally stored. This allows them to work independently. The network is used to transfer only user-to-user changes. Another reason for storing the trees locally is to prevent users from deleting someone else's nodes thus preventing to prevent potential errors. If one user deletes, moves or changes nodes, the other users can see them in the collaboration conflicts window (discussed below). To enable synchronous mode, the user needs to check the "Synchronized Collaboration Mode" button in the "Prefs" tab menu (Figure 7-4).

Imagine John and Sarah are creating another AI behavior for a game object called *Slime* in a game development jam working from home. Figure 7-2 shows *Slime* in the game scene chasing the player. They have an overall idea about their AI character in

games. Like earlier, this AI character has the same distinct states: patrolling, attacking and fleeing. However, how the AI character changes its state is different from before as follows:

1. If the guard (*Slime*) sees the player, the guard attacks.
2. If the guarded character (*Beholder)* is asking for help, the guard attacks.
3. If the guard is attacking but no longer seeing the opponent, the guard goes back to patrolling.
4. If the guard is attacking but is badly hurt, the guard starts fleeing to its home position to regain its full health.

In Figure 7-9, a white rectangular outline and user's name are used to differentiate the nodes that were added by John and Sarah. John first starts with the third condition from the above. To perform these actions, he adds a selector node (→), an interrupt node, an action node and assigns all the necessary tasks to these nodes (Figure 7-9). In the synchronous mode, all these nodes automatically appear on Sarah's canvas in real time. Sarah starts working on the attacking part of the fourth condition mentioned above. To perform this action, she adds a timeout node. Doing so results in it appearing on John's canvas in real time. Furthermore, Sarah adds a binary selector node (IF $HEALTH > 5), two action nodes (IN SEQUENCE .SEEK PLAYER .$HEALTH − =1 PER SECOND .OWNER.PLAYERDAMAGE() and IN SEQUENCE .SEEK $HOME .$HEALTH = 10). See the "Sarah" group in Figure 7-9. While this is being done, John is also working on the third branch of this behavior tree (see the "John" group on the right in Figure 7-9) where he completes the second condition from the list above. When everything is done, the resultant BT appears in Figure 7-9.

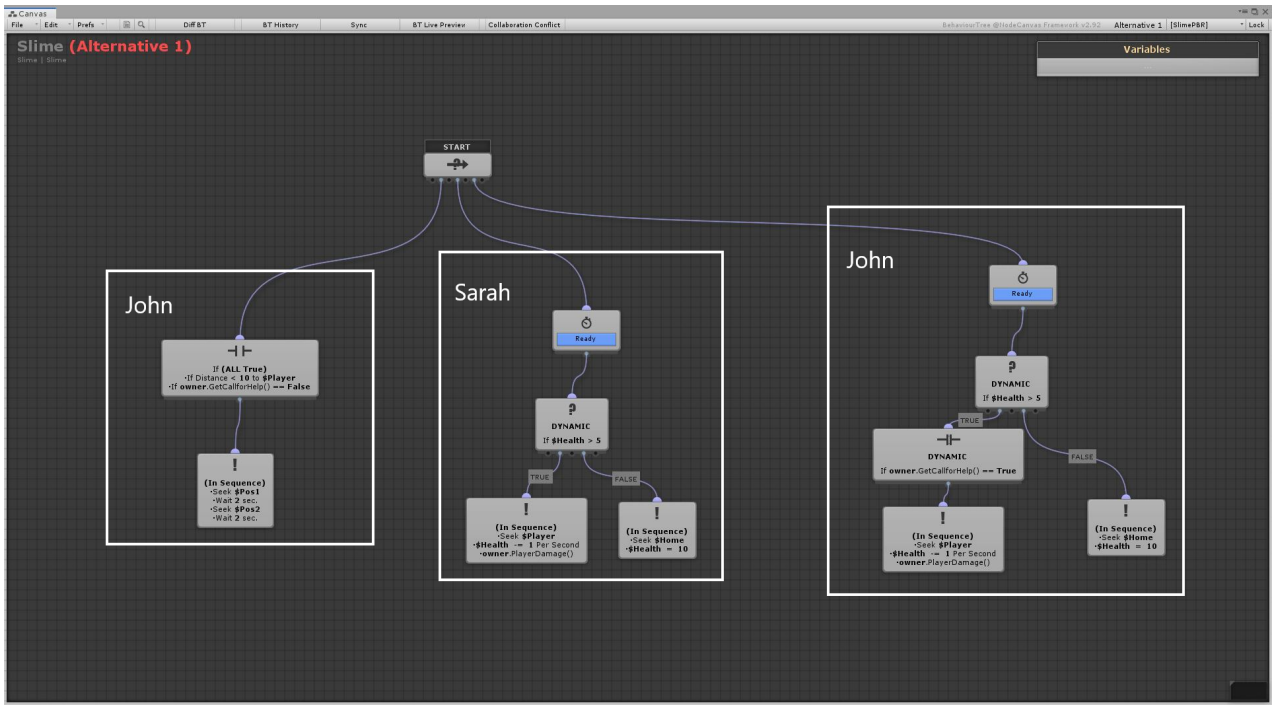


Figure 7-9: Resultant *Slime* BT after syncing John (J) and Sarah (S).

Sarah then decides to make some changes in the patrolling state to enable walking and idle animations. So, she deletes the action node (IN SEQUENCE, SEEK $POS1, SEEK, WAIT 2 SEC, $POS2, WAIT 2 SEC). In its place, Sarah adds a sequencer node and four action nodes and assigns necessary action tasks to them as shown using a highlighted white outline with the letter "Sarah" in Figure 7-10. Sarah could make changes to her BT. However, these changes do not affect John's BT because *NCCollab* was not designed to allow deletion of other users' nodes. This was implemented to prevent potential loss of important work of other users (only added nodes appear in other users' BTs). John can see these changes in the collaboration conflicts window and the difference visualization view (both discussed below). If John is not happy with the changes, he can use the "BT history" window (discussed below) and restore from previous states of the nodes from the history.

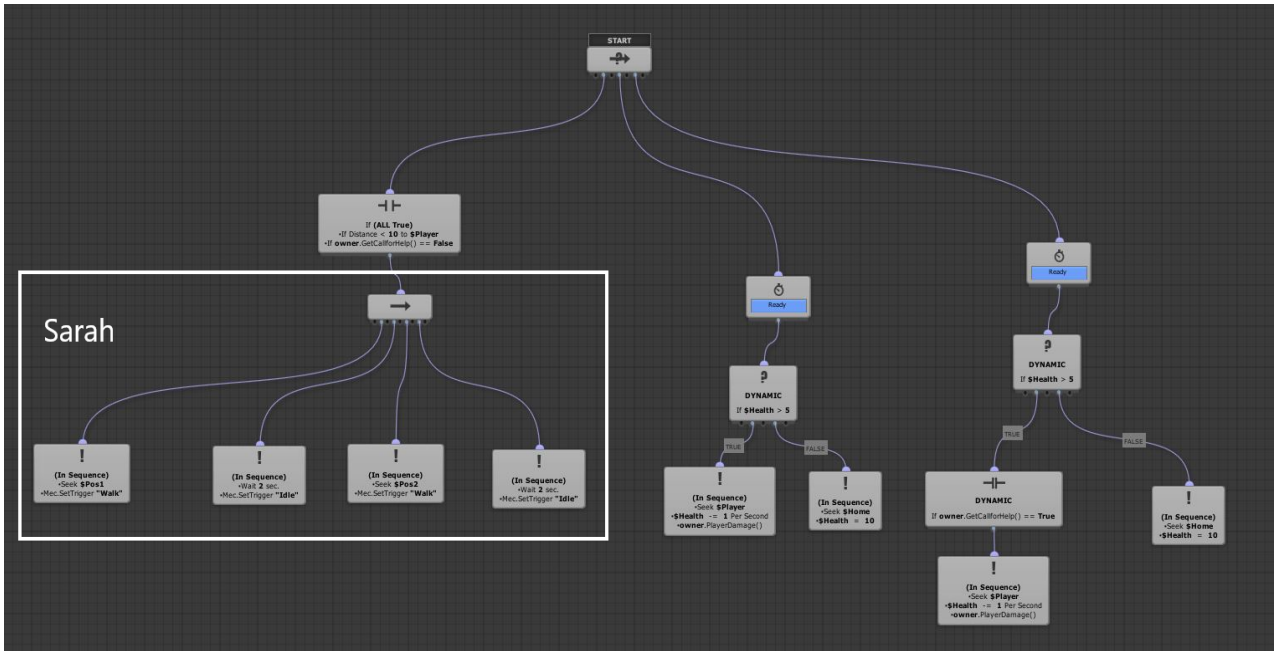


Figure 7-10. Sarah's *Slime* BT after enabling walking and idle animation.

7.7 Live Preview of BT Canvas

Live programming is a technique where programmers re-execute a program *continuously* while editing [102]. Live preview is a variation of this technique that refreshes the output immediately upon a change to the code, and it is ideally suited for UI-heavy application development such as visual programming. In a visual programming environment, this provides an immediate connection between the program's logic and the output for the developers so they can see the effects of changes to their tree or node in real time. This feature leads to fewer iterations of the AI development cycle, expedites the development process overall.

NCCollab provides a live preview of other collaborators' BTs in another canvas. This helps the user to have a clear idea on what the other collaborators are working on so that they can give them fast feedback. This also allows the user to guide other collaborators who are stuck by showing a live preview of their own BT. A live preview of John's *TurtleShell* BT is shown in Figure 7-11. To open this live preview, the user

needs to click on “BT Live Preview” (Figure 7-8) from the main canvas toolbar. By default, when the user opens the BT live preview, *NCCollab* automatically re-focuses on the center of the whole graph. This has been done by constructing a rectangle that encapsulates all the positions of the nodes. The rectangle consists of the following points: xMin, xMax, yMin, and yMax. *NCCollab* finds the center of this rectangle and focuses to this position. The background of the canvas color appears green instead of grey to allow the user better differentiate live preview from their working canvas. *NCCollab* shows the user’s name, AI character’s name and the name of the BT alternative on the top left of the live preview to give a clear idea about the opened BT. The live preview shows what John’s canvas looks like in real time except for the zoom and pane of the graph and the positions of the nodes. Since the users can resize the live preview window, changing zoom and pan can be necessary depending on the size of the window.

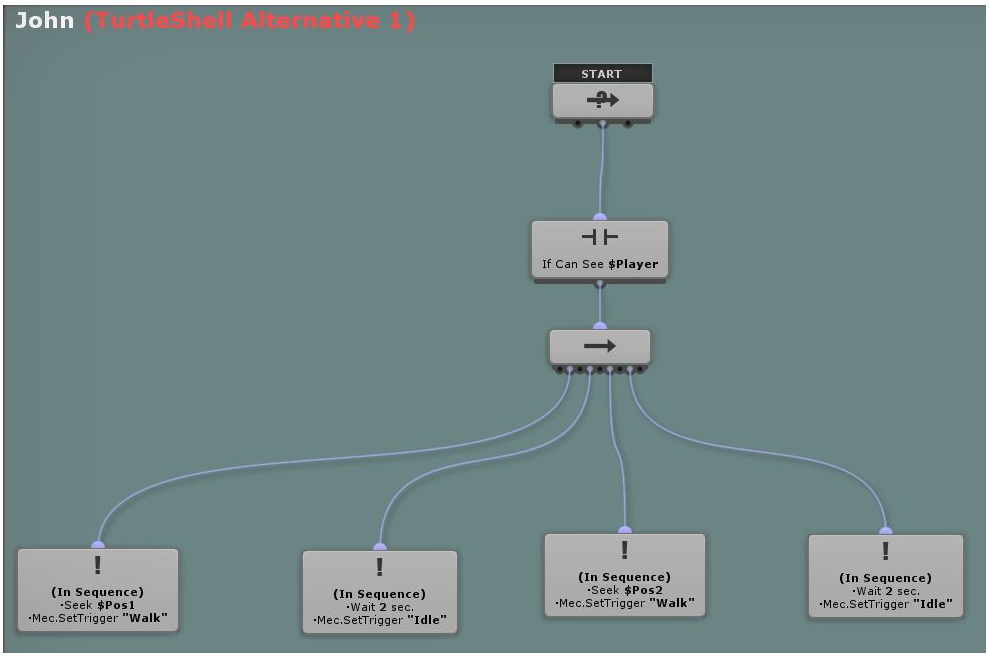


Figure 7-11: Live Preview of John's BT of the *TurtleShell* AI Character.

7.8 Conflict Resolution

In *NCCollab*, conflicts can arise when multiple users modify the properties of the same node or when operations are applied by one user on a node which was deleted by another user or in a situation when a user adds nodes that were never added by another user (only applicable for asynchronous collaboration). In these situations, the user can resolve the conflicts manually, similar to how it is done in version control systems like *Git* [39]. To see the conflicts the user needs to click on the Conflict Resolution GUI button from the *NCCollab* toolbar as shown in Figure 7-8. A conflict resolution window then appears where the information is displayed about the type of conflict, who it was caused by, the timestamp, and details. See Figure 7-15.

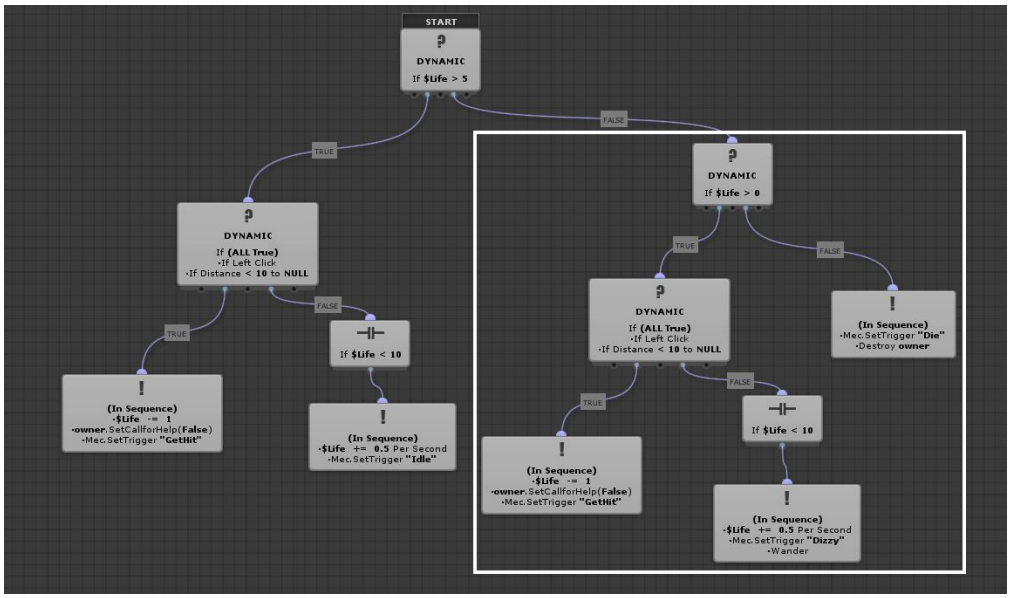


Figure 7-12. John's *Beholder* BT with both states completed.

Imagine, John and Sarah, are creating another AI behavior for a game object named *Beholder* in a game development jam while working from home. *Beholder* is guarded by *Slime* that was described earlier (see Synchronous Collaboration). above Section). *Beholder* has three states based on its health.

1. Idle State: *Beholder* remains in the idle state when its HEALTH > 5. If the player attacks the *Beholder*, it asks for help from *Slime*.
2. Dizzy state: *Beholder* stays in the dizzy state when the HEALTH <= 5, and it flees away.
3. Dead State: If the health of *Beholder* is below 0, it dies.

Suppose, John has completed developing all three states of *Beholder*'s BT (Figure 7-12). When Sarah syncs with John's BT only the left branch was completed, so she does not have access to the entire BT of *Beholder*. Sarah's *Beholder* BT is shown in Figure 7-13. After that, Sarah deleted one action node ($LIFE +=.5 PER SECOND .MEC.SETTRIGGER "IDLE") from the left branch (white box in Figure 7-13) and replaced it with another action node ($LIFE +=.5 PER SECOND SEEK POS1 .SEEK HOME) as shown in the white box of Figure 7-14. To see if there is a conflict, John opens the "Collaboration Conflicts" window as shown in Figure 7-15. He finds out that two

binary selectors nodes, two action nodes, one condition evaluator node (white box in Figure 7-12) have not been added by Sarah and one action node (IN SEQUENCE $LIFE=-1 PER SECOND MAC.SETTRIGGER "IDLE") has been deleted by Sarah and replaced with another action node (white box in Figure 7-14).

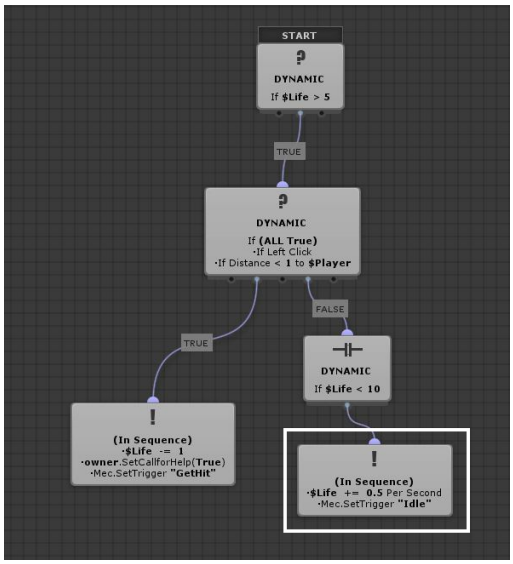


Figure 7-13. Sarah's *Beholder* BT with John's left branch only.

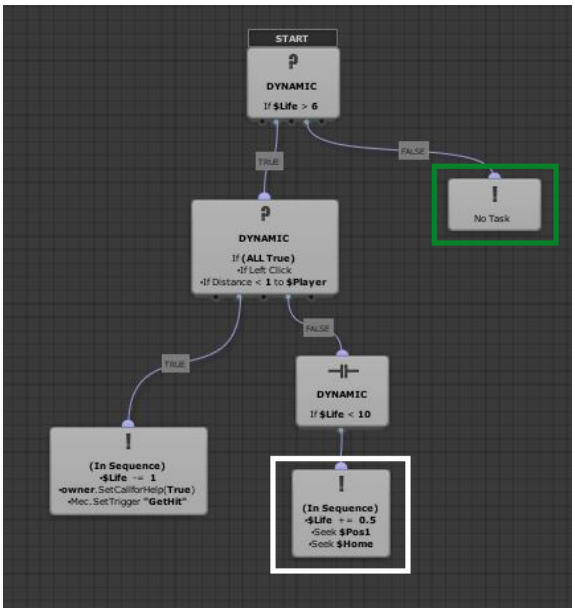


Figure 7-14 Sarah's *Beholder* BT with added (green rectangle) and replaced (white rectangle) node.

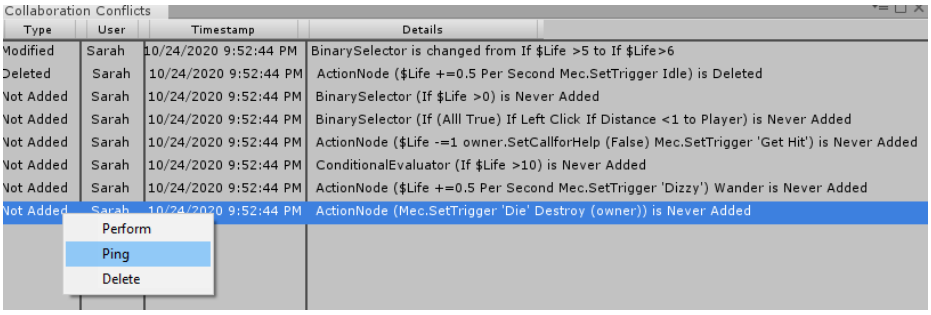


Figure 7-15. John's Collaboration Conflicts window using ping.

NCCollab uses a red and green rectangular outline to highlight nodes in the canvas that are involved in a conflict. E.g., to highlight a conflict in the BT, a user needs to right-click on a conflict and select "Ping". *NCCollab* then focuses on that node, highlights it using a red rectangular outline and flickers at a rate of 4 Hz between red and green for 2 seconds (Figure 7-16) to get the attention of the user.

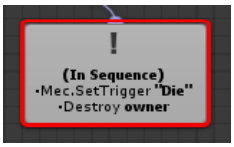


Figure 7-16 John's pinged node using Collaboration Conflicts window.

NCCollab offers partially automatic collaboration conflict resolution. It is partial automatic because the users do not need to manually modify or delete a node. We call it partial because the users need to choose manually which conflict they want to solve and automatic because they can select "Perform" to automatically resolve the selected conflicts. For example, in Figure 7-12 the prime node of John's *Beholder* BT is "IF $LIFE > 5" and in Figure 7-14 Sarah modified the prime node to "IF $LIFE > 6". So, to modify John's *Beholder* BT prime node, he needs to click on the "Perform" button as shown in Figure 7-17. *NCCollab* cannot resolve "Not Added" conflicts because one user cannot add nodes on another user's canvas. Although, if the users do not want to see the Not Added conflicts, they can delete those conflicts from the list.

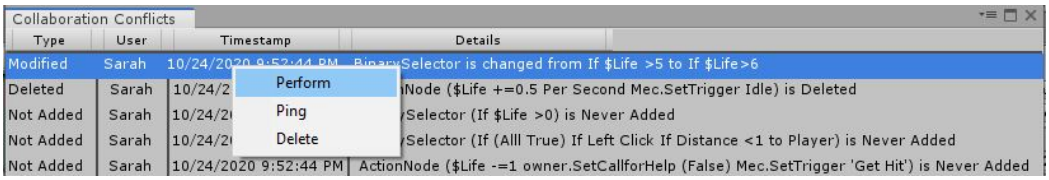


Figure 7-17 John's Collaboration Conflicts window using perform.

To delete a conflict from the list of collaboration conflicts, the user needs to click on the "Delete" button in Figure 7-17.

7.9 Difference Visualizations of BTs

Difference visualizations in *NCCollab* were adapted from *NCAlt*'s difference visualizations for alternatives. To achieve this, we have added a new type of difference visualization for *not added* nodes, which are the nodes that have not been added before. We also changed the way we visualize added, deleted, priority changed, and modified nodes based on the feedbacks that we have solicited from the participants of the user study on *NCAlt.*

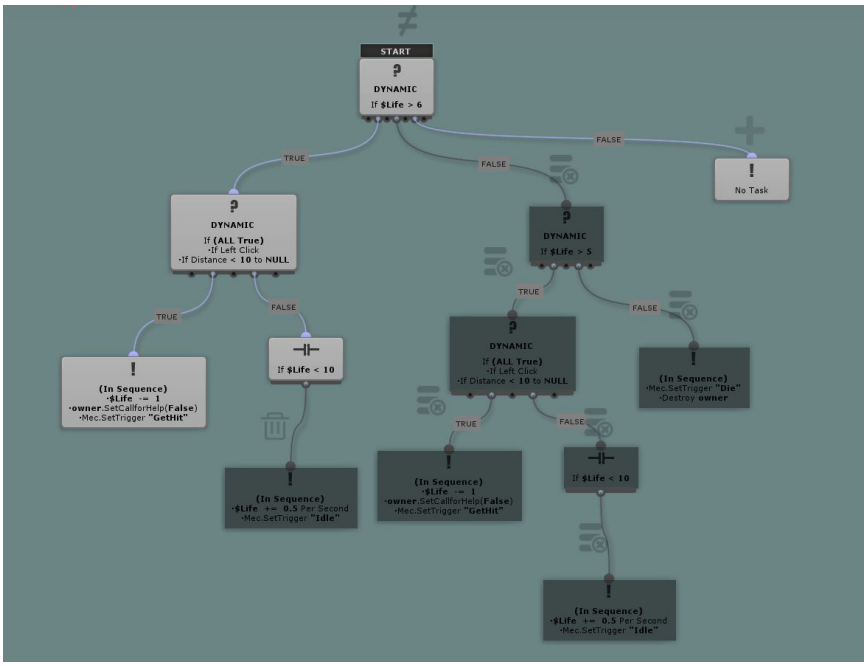


Figure 7-18. Difference visualization of the BT of *Beholder* AI Character with differences marked.

To show difference visualizations in the BT, we will use the same *Beholder'*s BT example that we have discussed above (see Conflict Resolution). In that example, John completed both states of the *Beholder*'s AI character (Figure 7-12). However, when Sarah synced with John's BT, only the left branch was completed, therefore she did not have access to the entire BT of *Beholder*'s AI character. Besides, Sarah deleted one action node from the left branch (white box in Figure 7-13), added one empty action node (green box in Figure 7-14) to the rightmost branch of the tree and modified the first binary selector node (IF $LIFE >6 in Figure 7-14). Therefore, when John compares his BT with Sarah's in the BT difference visualizations, he discovers the following differences listed below.

7.9.1 *Not Added nodes*

As shown in Figure 7-18, Sarah has six nodes (white box in Figure 7-12) relative to John's *Beholder* BT that were not added from before. A missing icon on top of the

nodes and highlighted as reduced transparency informs the user that they have not added these nodes from before.

7.9.2 *Deleted nodes*

As shown in Figure 7-18, Sarah has deleted one node (white box in Figure 7-13). A garbage bin icon on top and highlighted as reduced transparency informs the user that they deleted the nodes.

7.9.3 *Added Nodes*

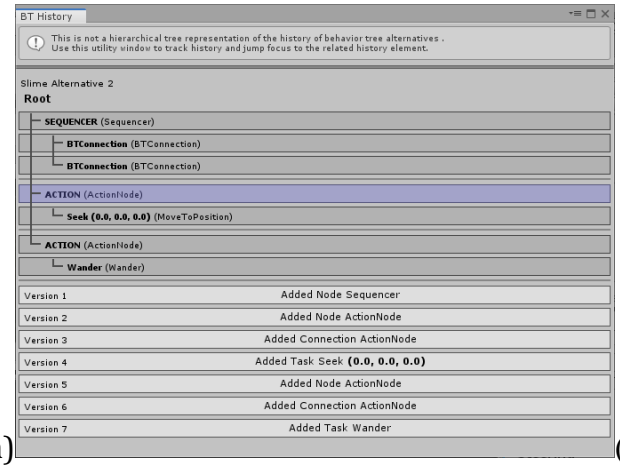
As shown in Figure 7-18, Sarah has added one empty action node. A "+" icon on top of the nodes informs the user that these nodes were newly added relative to John's *Beholder* BT.

Both deleted and not added nodes are visualized with reduced transparency using a complementary shade of grey for the dark green background color of the live preview. This approach of using reduced transparency was found to be effective in the previous work [120].

7.9.4 *Modified nodes*

As shown in Figure 7-18, Sarah has modified one Binary Selector node from $HEALTH >5 TO $HEALTH>6. A "≠" icon displaying on top of the nodes informs the user that these nodes were modified relative to John's *Beholder* BT.

7.10 BT History

(a) 

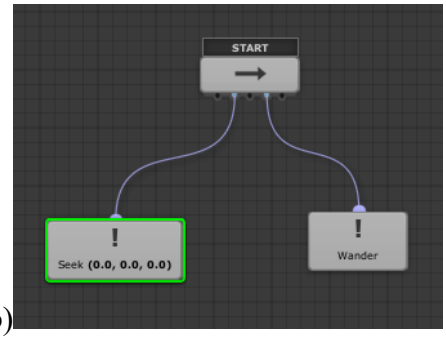
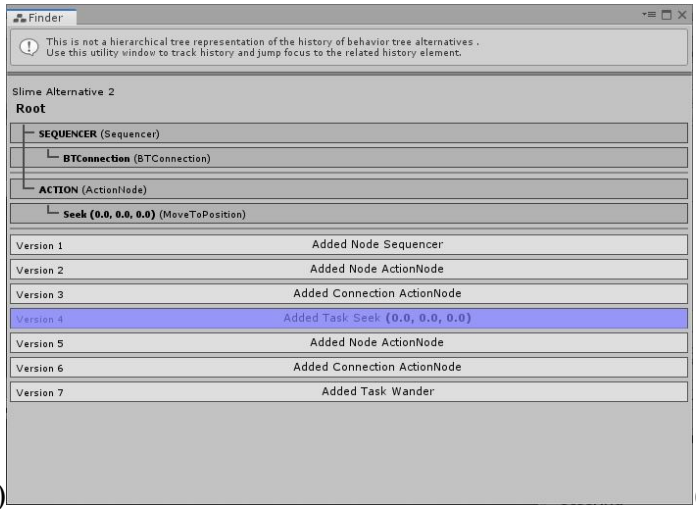
(b) 

Figure 7-19 (a) BT History window (b) Version 7 opened in the canvas.

The ability to restore from a previous state makes *NCCollab* a more complete versioning system by allowing the users to look through their past work and select a particular state of a BT. Previous states are accessible in *NCCollab* in the "BT History" window (Figure 7-19 (a)). BT history contains all the previous states of the BT. BT history is updated each time the user performs an operation on the BT. Whenever the user makes a change in the BT such as adding a node or a task, deleting a node or a task, or modifying a node or a task, a new state of the BT appears in the BT History (Figure 7-19 (a)). Figure 7-19 (b) contains the final version 7 of the behavior tree where a sequencer node and two action nodes have been added to the canvas. When the user selects version 4 from BT History (Figure 7-20 (a)), version 4 appears in the canvas (Figure 7-20 (b)). Version 4 contains a sequencer node and an action node with the "SEEK (0.0, 0.0, 0.0)" assigned task. The user can use the BT History as a version control tool. BT history is stored in a JSON file that contains a unique ID, an operation such as "Added Node Sequencer" or "Added Task Wander", and a filename of a JSON file that contains the entire BT of that version.

7.11 Hierarchical BT Representation

In the BT History window, BTs are shown using a hierarchical view of the tree. This hierarchical representation gives the user visual feedback in the BT History window without the need to look at the canvas for the selected version of the BT every time a change is made. *NCCollab* also helps to visualize a specific node or a connector from the hierarchy by highlighting them in the BT. When the user moves the mouse cursor on top of a node or a connector in the hierarchy, the node or a connect will flicker. In Figure 7-19 (a), the user hovers the mouse cursor above the action node in the hierarchy and in Figure 7-19 (b), the same action node is highlighted with green and flicks.

(a) 

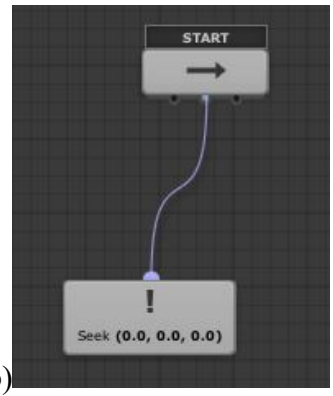
(b) 

Figure 7-20. (a) BT History window Version 4 selected (b) Version 4 hosted in the canvas.

7.12 Instant Messaging

The integration of communication features into collaborative tools can help collaborators discuss and resolve issues without losing focus on the code [101]. So, for collaborative environments, built-in instant messaging is found to be effective, see e.g., [2, 55, 101]. *NCCollab* offers a built-in instant messaging system where collaborators can leave notes or ask for each other's help. E.g., John is trying to get an update of a BT from Sarah. To start the text chat, John right-clicks on the empty area of the canvas to

trigger the context sensitive menu and clicks on “Send message to Sarah“ from the list as shown in Figure 7-21.

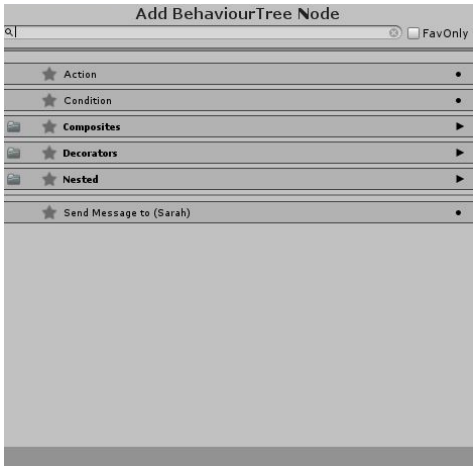


Figure 7-21. Accessing the chat box.

Then, the chat box appears (Figure 7-22) for the recipient user. Through this chat box John and Sarah can send each other instant text messages. Imagine, John sends Sarah the first message “Hi”, but Sarah does not have this chat box open. She receives an instant notification update as shown in Figure 7-23. Here, *NCCollab* follows the “continuous coordination” model introduced by van der Hoek et al. [57] where a system needs to notify the developers of events relevant to them. *NCCollab* sends an instant notification to the other user because the recipient’s chat window was not open at that time.

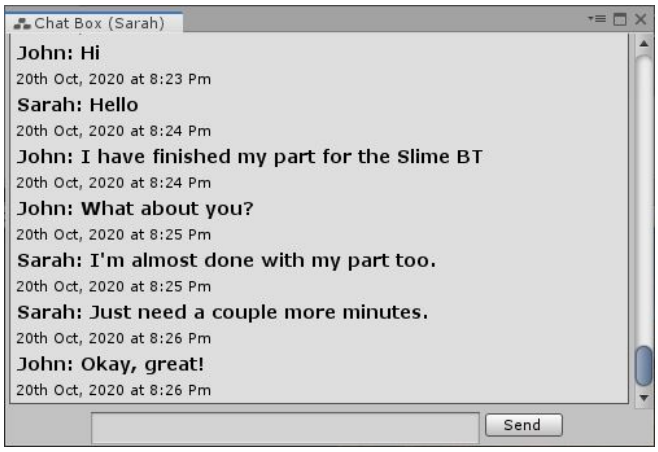


Figure 7-22. Accessing the chat window.

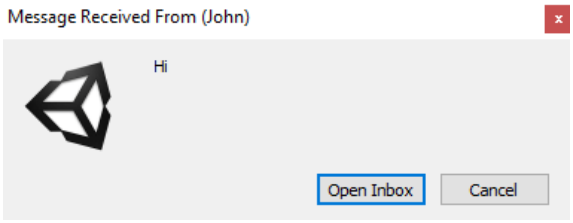


Figure 7-23. Message notification window.

Chapter 8

User Study II

In our user study described below, we mainly evaluated working on the same BT alternatives since users can use *GitHub* [13] or similar platforms to share the BT JSON file and collaborate. However, doing so would not be efficient compared to using *NCCollab* they can collaborate without an extra effort such as uploading and downloading BT JSON files manually.

We performed a user study, the goals of which were to:

1. compare synchronous and synchronous modes in *NCCollab* against *NodeCanvas* (the control condition where none of the collaboration support was available);
2. study the usability and usefulness of
    a. BT history for restoring previous states of BT,
    b. live view of the BT,
    c. conflict resolution, and
    d. integrated instant messaging;
3. investigate how different collaborative modes affect collaboration, exploration, confidence, chance of future use; and
4. the overall process.

8.1 Participants

We recruited 12 paid participants (ten males, one female and one preferred not to say) from relevant technical undergraduate & graduate programs and from among industry professionals. The participants' backgrounds were game development and computer science. All the participants were between 21 and 35 years old (M = 24.6, SD = 2.6) and had on average 17.3 years of experience using a desktop/laptop computer (SD = 5.3,

years). All the participants had between 2 and 10 years of experience with *Unity* (*M* = 5.3, *SD* = 2.4). Six participants had on average 1.3 years (*SD* = 1.03) of experience with *Unreal Engine*. All participants on average had 2.1 (*SD* = 1.8) years of experience with at least one visual scripting system. Seven participants previously worked with *Unreal Engine Blueprints*, three — with *Blender Node Editor*, two — with *Flow Canvas*, two — with *Nodebox,* two — with *NodeCanvas,* one — with *Maya,* one — with *Blackmagic Fusion* and one — with *Nuke*. Some participants also had experience working with block-based system programming languages as follows: four — with *Scratch,* two — with *Blocky,* two — with *Snap!,* one — with *Alice,* and one — with *GML (Game Maker Language)*. We also asked participants to rate their level of proficiency with general-purpose programming languages, such as C, C++, C#, Java, Python, Kotlin, Go, Swift etc. Three participants had average proficiency, six — above average, and another three — professional level proficiency. Two participants took an AI course where they studied behavior trees.

8.1.1 *Recruitment*

Convenience sampling was employed for recruitment. The recruitment process has occurred through the university's Game Development program's semi-official *Discord* (an online communication tool similar to Slack) and sending an individual email to the participants who participated in the previous study for *NCAlt*. This *Discord* channel includes present and past undergraduate Game Development students as well as all the graduate Computer Science students in HCI and related fields. The messages have indicated that interested students can sign up for a timeslot using an online tool called *Calendly*. Participants were told they should not publicly respond to the *Discord* message on the *Discord* channel, to ensure confidentiality. The consent form was also

attached to the message so that potential participants would be able to review it before deciding to volunteer.

8.2 Apparatus

Since this study was performed online due to COVID-19 pandemic, we collected specifications of the PCs participants used to run the study, which appear below.

8.2.1 *CPU*

Five participants had Intel *Core-i7* (8 core) processor, two had AMD *Ryzen* 7 (8 core) processor, two had AMD *Ryzen* 5 (6 core) processor, two had AMD *FX* (8 core) processor and one had Intel *Core-i5* (6 cores) processor.

8.2.2 *GPU*

Four participants had NVIDIA *RTX* 2060, two participants had NVIDIA *GTX 1050*, two participants had NVIDIA *GTX 1070*, one participant had NVIDIA *GTX 2070*, one participant had Intel UHD *Graphics 630*, one participant had Asus *Strix 2080*, one participant had EVGA *2070 Super*.

8.2.3 *RAM*

Eight participants had 16 GB of RAM on their PC, three had 32 GB and one had 8 GB.

8.2.4 *Other Hardware & Software*

Four of the participants had a dual monitor setup, so they kept all the *NCCollab* windows on one monitor and the other general *Unity* windows (Scene, Game tab, Hierarchy, Inspector, etc.) on another monitor. We did not collect information about the operating systems that participants they were using during the study on their PCs. However, we tested *NCCollab* on Microsoft *Windows* 10, *macOS* 10.15 and Linux-

based operating systems prior to the study. Our tests revealed that *NCCollab* runs the same way without noticeable differences in all three operating systems.

8.3 Procedure

8.3.1 *Tutorials and Tasks*

We designed three tasks for the study. One task was performed using *NodeCanvas* (NC), which was the control condition. Another task was performed using the asynchronous mode in *NCCollab*. The third task was performed using the synchronous mode in *NCCollab*. Participants were given a *Unity* game project template that contained all the game objects and the game scenes for all three tasks. Participants were asked to create an AI for three specific game objects (*Slime*, *TurtleShell*, and *Beholder*). Each of the tasks was preceded by a written tutorial where the corresponding task and template were thoroughly explained. For *NC* the tutorial covered how to create these different types of nodes and connect them and how to write a script for a node. For *NCCollab* the tutorial also covered how to use synchronous and asynchronous versions of *NCCollab*, versioning of BTs through restoring from BT history window, live BT window to see another participant's canvas, solving conflicts in the behavior tree, and sending text messages through the chat window. The participants were asked to perform a warm-up task as shown in the tutorial so that they are comfortable to use the system when they started the actual tasks.

For the actual tasks, participants were asked to create three behaviors for three game objects in an RPG game. Having three different games for three different tasks would have been ideal, but that would make the study too long as it would require more learning on the part of the participants. So, we decided to use only one game. In *NCCollab*, the participants were only told to use the synchronous or asynchronous

modes for two tasks. However, we did not impose the use of the auxiliary features: BT history, live BT window, conflict resolution, and instant messaging. As a result, the participants had full freedom whether they wanted to use the auxiliary features or not. Giving this freedom allowed us to investigate how and in which scenarios the participants would use a certain auxiliary feature in *NCCollab*.

In the control condition (None), the participants were told to create an AI for the game object offline, without the use of collaboration. This also meant did not communicate with each other and did not coordinate their work in any way. In this condition, they used the vanilla version of *NodeCanvas*. The three tasks involved developing AI behaviors for three AI characters in a game. The tasks were inspired by examples 1 and 2 provided by Richard Moss [76] and by following the guidelines provided by Ben Sizer [106]. The tasks were similar to what we described in the worked example with John and Sarah. All the AI characters' BTs are shown in Figure 7-8, Figure 7-9, and Figure 7-12. The movement of the humanoid player character was pre-written where the character could:

- move in any direction by using the arrow or WASD keys on the keyboard;
- attack by pressing the left button of the mouse;
- rotate the camera around the player by holding the left mouse button, and
- zoom in and out by scrolling the mouse.

Participants needed to work on the AIs for *Slime*, *TurtleShell*, and *Beholder*. Three functions for the AIs were pre-written in the script. These included *playerdamage, SetCallforHelp,* and *CheckCallforHelp*. For all three AI characters, an empty BT was created for them, so the participants had to make them BTs from scratch.

8.3.2 *Experimental Design, Independent & Dependent Variables, and Data Collection*

The study used a 6 × 3 mixed factorial design. See Table 8-1 for details.

Independent Variable	**Type**	**Levels**
Order	Between-subject	A. Sync→ Async→None; B. Async→ Sync→ None; C. None → Async→ Sync; D. None → Sync→ Async; E. Async → None → Sync; F. Sync → None → Async.
Mode	Within-subject	None, Async, Sync

Table 8-1. Independent variables and levels used in the study.

To minimize the effect of a potential confounding variable of pairing certain tasks with certain modes, the pairings were randomized without replacement according to Table 8-2.

Participant #	**Task 1**	**Task 2**	**Task 3**
P1, P2	*Slime* with None	*TurtleShell* with Async	*Beholder* with Sync
P3, P4	*Slime* with Async	*TurtleShell* with Sync	*Beholder* with None
P5, P6	*Slime* with Sync	*TurtleShell* with None	*Beholder* with Async
P7, P8	*Slime* with Sync	*TurtleShell* with Async	*Beholder* with None
P9, P10	*Slime* with None	*TurtleShell* with Sync	*Beholder* with Async
P11, P12	*Slime* with Async	*TurtleShell* with *None*	*Beholder* with Sync

Table 8-2. Randomization of task and mode pairings between participants.

Creativity Support Index (CSI) [19] is a quantitative psychometric survey that measures how well creativity the design process is supported by a tool. In this survey, participants rate six dimensions of creativity support: Enjoyment, Exploration,

Expressiveness, Immersion, Results Worth Effort, and Collaboration. We used this survey to measure participants' CSI scores for each mode (None, Async, Sync). This survey is particularly well-suited for rating the extent of how well a certain system supports collaboration. The CSI score was one of the dependent variables.

NASA-TLX is a multi-dimensional scale designed to obtain workload estimates from one or more operators while they are performing a task or immediately afterwards [52, 53]. The CSI was inspired by and modeled after NASA-TLX [19]. Galy et al. [38] propose a method of analyzing the gathered NASA-TLX data, which is to analyze the individual subscales. Analogously, we analyzed individual weighted factor scores from the CSI survey that we administrated. As a result, the weighted factor score was another dependent variable.

Finally, we logged how long it took participants to complete each task in minutes. Task completion time is also another dependent variable in this study.

8.4 Quantitative Results

8.4.1 *Creativity Support Index*

The breakdown of CSI results is shown in Table 8-3.

Mode	Factor/ Scale	Collaboration	Enjoyment	Exploration	Expressiveness	Immersion	Results Worth Effort
	Factor counts (SD)	3.6 (0.7)	1.8 (0.9)	3.5 (0.6)	2.2 (0.8)	1.6 (0.6)	3.8 (1.3)
	Factor Score (SD)	0 (0)	12.9 (1.9)	12.7 (2.3)	13.1 (2.0)	10.7 (1.5)	14.4 (1.7)

None	Weighted Factor Score (SD)	0 (0)	22.5 (10.7)	45.6 (17.7)	29.6 (16.4)	17.5 (6.6)	53.0 (15.9)
Async	Factor counts (SD)	3.6 (0.7)	1.8 (0.9)	3.5 (0.6)	2.2 (0.8)	1.6 (0.6)	3.8 (1.3)
	Factor Score (SD)	15.5 (1.5)	14.0 (1.9)	14.3 (1.3)	13.4 (1.6)	10.9 (1.8)	16.1 (0.9)
	Weighted Factor Score (SD)	55.6 (11.1)	23.7 (9.4)	50.4 (13.4)	30.2 (15.7)	18.2 (8.7)	61.2 (20.6)
Sync	Factor counts (SD)	3.6 (0.7)	1.8 (0.9)	3.5 (0.6)	2.2 (0.8)	1.6 (0.6)	3.8 (1.3)
	Factor Score (SD)	13.0 (1.8)	12.8 (2.2)	12.9 (2.1)	12.8 (2.2)	10.9 (2.0)	14.6 (2.4)
	Weighted Factor Score (SD)	46.6 (11.2)	21.6 (9.3)	46.2 (17.1)	28.1 (14.1)	17.1 (7.8)	54.3 (21.1)

Table 8-3 CSI averages for all three modes.

The main effect of Order was not significant, $F(5,6) = .93$, ns, indicating that counterbalancing was successful. A Mauchly's test for sphericity revealed that sphericity was violated for Mode, $W = 0.256$, $p < .05$. The degrees of freedom were corrected using Greenhouse-Geisser estimates of sphericity ($\varepsilon = .573$). The main effect of Mode was significant, $F(1.14,6.88) = 26.99$, $p < .001$, $\eta^2_P = .81$. A pairwise post-hoc t-test with Bonferroni correction revealed that the mean CSI score for None ($M = 57.36$, $SD = 11.50$) was different from both: Async ($M = 77.59$, $SD = 8.99$), $p < .0001$ and

Sync (M = 72.27, SD = 12.86), $p < .01$. There was no significant difference between Async and Sync, $p > .05$. See Figure 8-1.

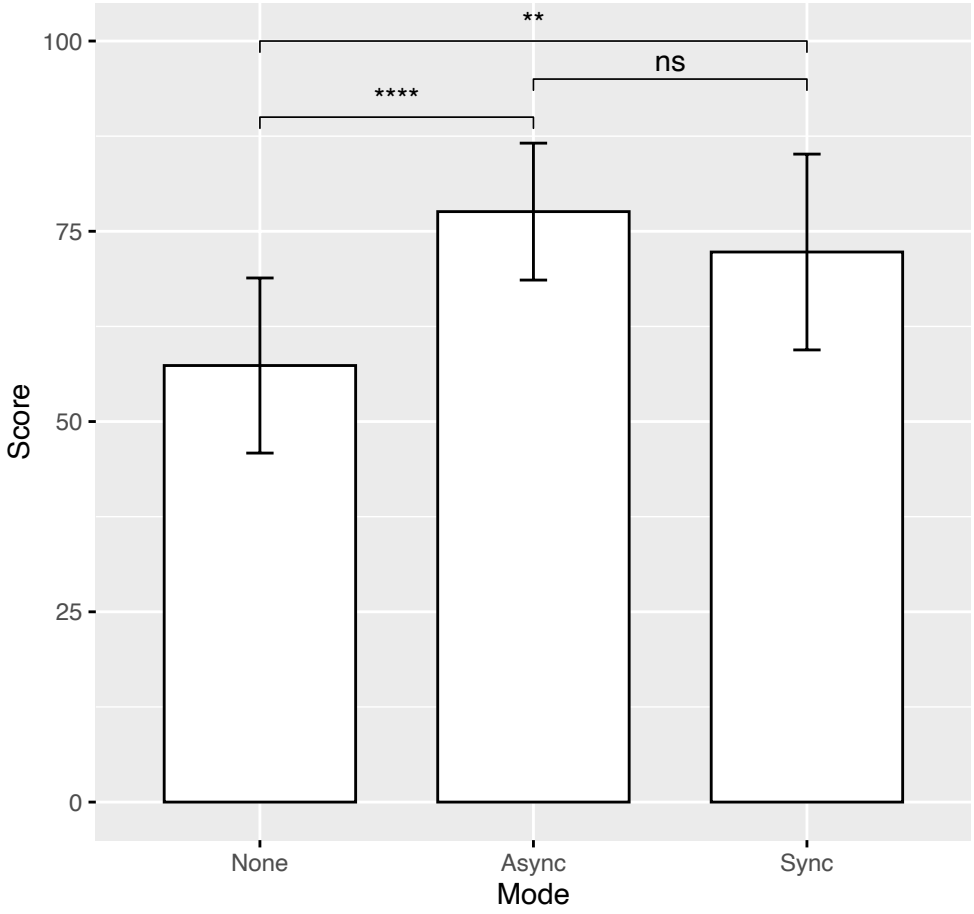


Figure 8-1. Mean CSI scores for the two collaboration modes and the control with significance levels from the post-hoc pairwise t-test with Bonferroni correction. Error bars: ±1 SD. ns: $p > .05$, **: $p < .01$, ****: $p < .0001$.

8.4.2 *Weighted Factor Scores*

We ran mixed ANOVA on weighted factor scores for Enjoyment, Exploration, Expressiveness, Impression, and Results Worth Effort. None of the results were significant. Since collaboration was not rated for None, we ran a dependent t-test, which revealed that Async (M = 55.5, SD = 11.5) was significantly different from Sync (M = 46.58, SD = 11.66), $t(11) = 3.5975$, $p < .01$, $d = 0.77$.

8.4.3 *Self-Developed Questionnaire*

We asked the participants to rate None as well as Async and Sync modes in *NCCollab* on a Likert scale from 1 (lowest) to 7 (highest) for confidence, efficiency, chance of future use. In Async and Sync modes only, we also asked the participants to rate the features of live BT preview, BT history, conflict resolution, and instant messaging. Finally, all modes were also rated overall. A divergent bar chart summarizes the results in Figure 8-2.

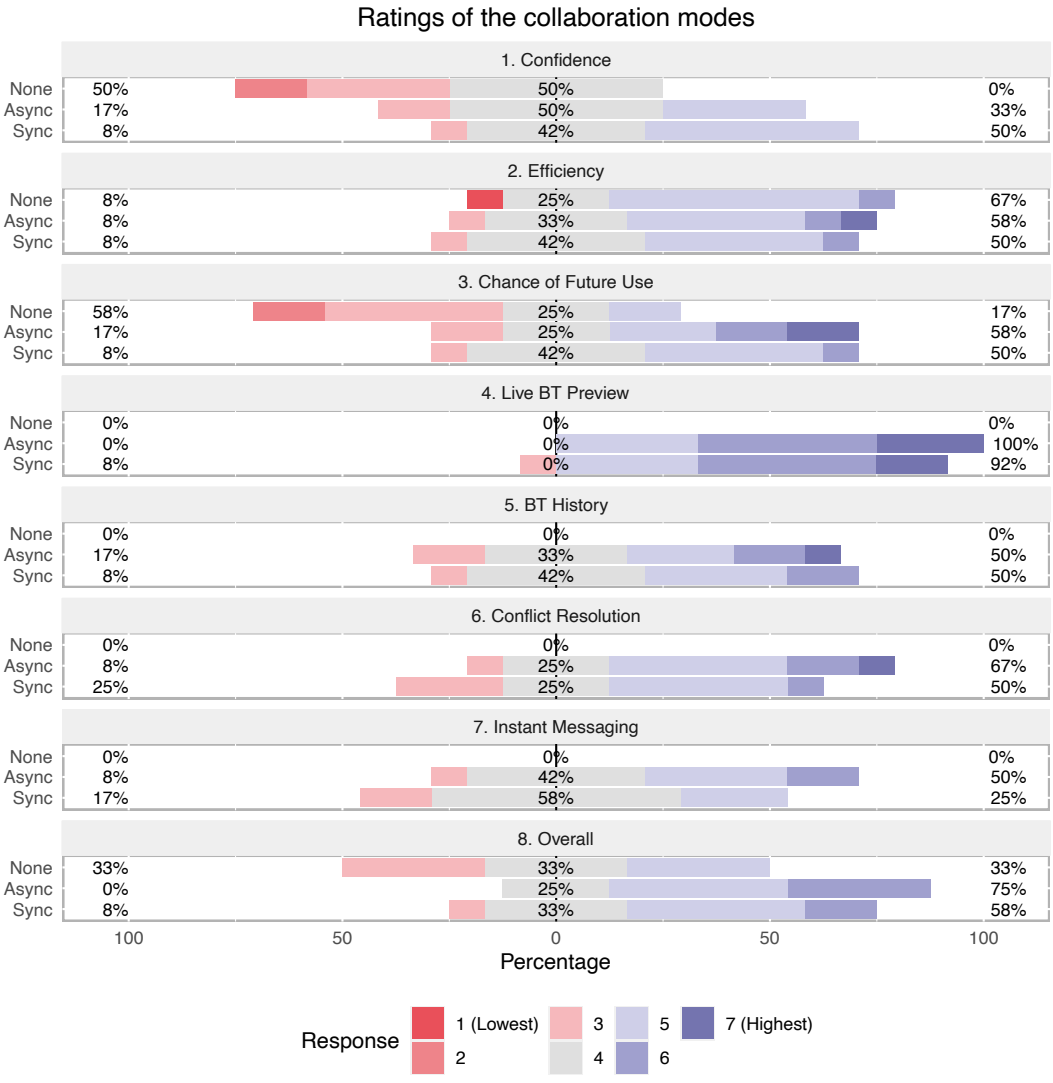


Figure 8-2. A diverging stacked bar chart comparing the ratings of the two collaboration modes (Async and Sync) and a None.

A Friedman rank sum test revealed a significant difference for Confidence, $\chi^2(2) = 14.774$, $p < .001$, $W = .615$. A post-hoc Conover test further revealed that Async (*Mdn* = 4) was different from None (*Mdn* = 3.5), $p < .05$, and that Sync (*Mdn* = 4.5) was different from None, $p < .05$. A Friedman rank sum test also revealed a significant difference for Chance of Future Use, $\chi^2(2) = 9$, $p < .05$, $W = .375$. A post-hoc Conover test further revealed that Sync (*Mdn* = 4.5) was different from None (*Mdn* = 3), $p < .05$. All other differences were not significant.

8.4.4 *Task Completion Time*

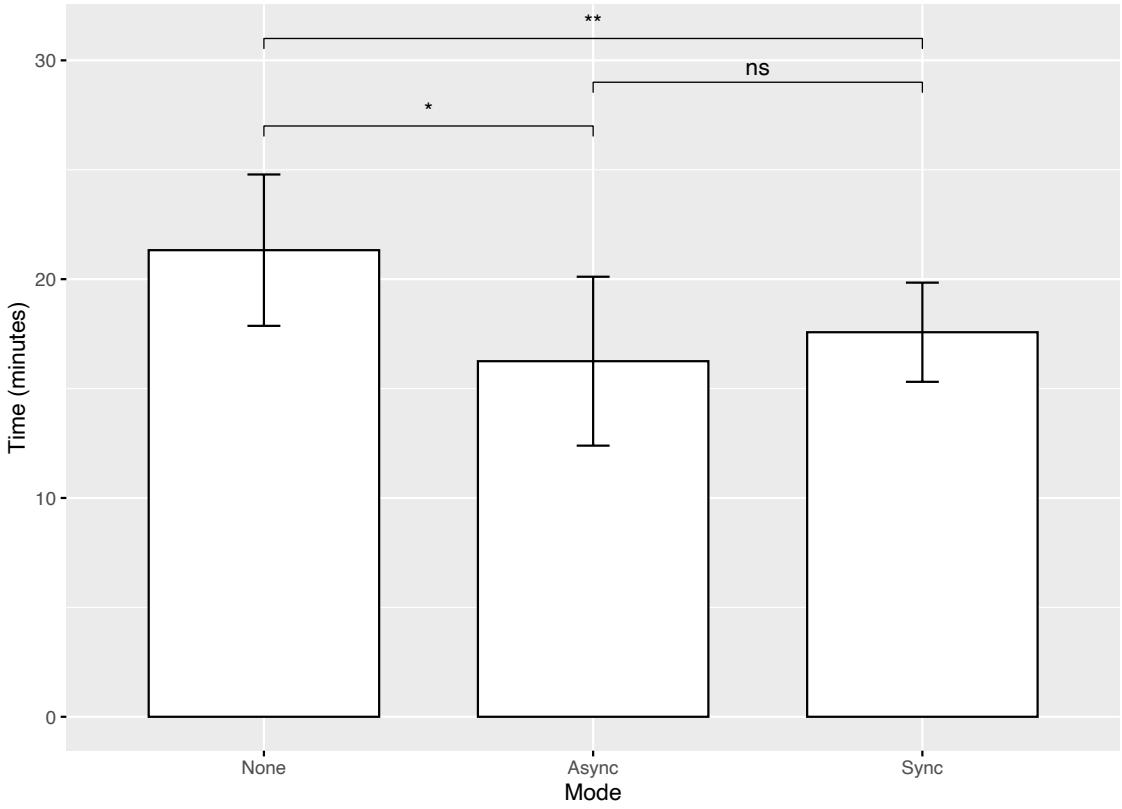


Figure 8-3. Mean task completion times for the three collaboration modes with significance levels from the post-hoc pairwise t-test with Bonferroni correction. Error bars: ±1 SD. ns: $p > .05$, *: $p < .05$, **: $p < .01$.

The main effect of Order was not significant, $F(5,6) = 0.64$, ns, indicating that counterbalancing was successful. The main effect of Mode was significant, $F(2,12) = 5.52$, $p < .05$, $\eta^2_P = .48$. A pairwise post-hoc t-test with Bonferroni correction revealed

that the mean task completion time for None (M = 21.32, SD = 3.5) was different from both: Async (M = 16.25, SD = 3.85), $p < .05$ and Sync (M = 17.57, SD = 2.26), $p < .01$. There was no significant difference between Async and Sync, $p > .05$.

8.5 Qualitative Results

We administered a semi-structured interview with the participants at the end of the study. During this time, the participants were asked to express their opinions, to provide feedback on the overall experience, and to explain the reasoning behind their decisions.

8.5.1 *Live Preview*

Overall, live preview was one of the most popular features of *NCCollab*, about which nearly all the participants felt positive. This is likely because live preview enabled them to be instantly aware about the updates their partners made to their BTs. P4 stated: "*I really enjoyed the live preview feature, being able to see what my peer was working on*". P2 stated: "*The collaborative features seemed fairly robust, I could preview what was happening over the network and the chat window enhances the collaboration*". These qualitative findings agree with the results of the self-developed Likert questionnaire in Figure 8-2 for live preview. Moreover, P10 suggested that live preview could be also used for teaching and learning.

8.5.2 *BT History*

Arguably, the addition of BT history brings *NCCollab* one step closer to being a fully-fledged version control system like *Git* [39]. All the participants were familiar with the idea of version control systems such as *Git*. The participants liked the idea that *NCCollab*, to an extent, offers similar functionality through BT history. P6 stated: *"I like that it's like a live version control. It has lots of analogs to programming and version control making it easier to understand".* Although P3, P5 mentioned that they

did not use the BT history in this study because they were too focused on completing the task, they did see the benefit of having this feature in the long run. They thought that they would be more willing to use BT history if they were working on a project with a larger scope. P5 suggested an interesting use case for BT history: after a long break, if the developers cannot exactly remember which part they were working on, they can just restore BT from history to see the changes they have made before the break. These findings somewhat agree with the results from the self-developed questionnaire in Figure 8-2 for BT history, since half felt positive about the feature in both Sync and Async modes, while less than 20% felt negative. P12 did not end up using BT history because they stated that "*I would [be] more willing to use the history tracking feature if I was working on an open-ended task or a larger BT*".

8.5.3 *BT Hierarchy*

We asked the participants if they found it useful to see a BT represented as a hierarchy. P7 found it useful because they liked simple text-based representation in the form of a hierarchy. However, P12 did not find the hierarchy that beneficial because the original BT structure was already straight forward.

8.5.4 *Collaboration*

We asked the participants what their preferred mode of collaboration was (None, Async, or Sync). Eight of them were in favor of Async and four were in favor of Sync. P9 stated that they were missing the collaboration features when the vanilla N*odeCanvas* was presented to two collaborative modes "*I did miss having some of the collaboration features such as the syncing when using this variant. It wasn't necessarily something I disliked but it felt like something was missing without them".*

8.5.5 *Instant Messaging*

We were observing the instances when the participants used instant messaging during the task. The following purposes of use have been identified as a pattern:

i) to divide their work on the behavior tree at the start,

ii) to check if another collaborator can see their update in real time,

iii) to signal to the other collaborator that the task has been completed or to confirm this.

In one instance, after completing one of the tasks, P7 sent a message to P8 but since *NCCollab* does not use sound to notify the user, P8 never noticed that message. So, P7 had to send multiple messages. P7 stated "*I would want the receiver to get a sound notification after receiving every new message*". We also asked all the participants how could we improve the usability of the messaging. P8 stated "*I would prefer voice chat over text chat*".

8.5.6 *Usefulness of NCCollab*

We asked the participants regarding the situation where the participants would use *NCCollab*. P3, P4, P7, P11 stated that since there is a time limit, *NCCollab* would be useful in a game jam. P4 stated "*I would like to use NCCollab synchronous mode in a game jam where I need to make a prototype of a game in a short time*". Coincidentally, we have also used this game jam scenario to describe the features of *NCCollab* above. Moreover. P8 mentioned three noteworthy use case scenarios for *NCCollab* as follows:

(i) Use by novice users such as design professionals,

(ii) Use by two experienced developers,

(iii) Use by one experienced and one novice developer.

8.5.7 *Game Genres*

We were interested to learn for developing which types of games *NCCollab* would be most useful according to the participants. P4 suggested it would be useful for prototyping, P9 suggested third-person role-playing games. However, P6 suggested that regardless if collaboration is supported or not, *NodeCanvas* would not work for puzzle games. Although, it is already expected because the main purpose of *NodeCanvas* is to develop the AI of a character, which these genres do not utilize.

8.6 Criticism, Suggestions & Uncovered Issues

We received some useful feedback on how to improve the system further. P6 had an issue with getting lost between two canvas windows because one was the main canvas that they were working on and the other was showing live preview of another user's BT. So, P6 suggested to fit both canvases in one window so that the user can see both at the same time and resize live BT preview. P1 suggested adding a feature of locking a node so that the other users cannot make any changes to them. Moreover, P1 also suggested to have some sort of visual representation to show which node is selected or being edited by another user.

P7 suggested having an optional comment field in the BT history window beside every state, similar to how *Git* [39] has the option to add a comment on a change to provide details. P3 and P12 had a problem with instant messaging, as the messages would not wrap. So, in order to see long messages, they had to make the window wider. P5 also noticed that they could not press the enter key on the keyboard to send a message and was forced to click on the send button, which slowed down their chatting experience.

Chapter 9
Discussion of User Study II

Here we describe and interpret the significance of our findings in the light of *NCCollab*. Methodological triangulation provides multiple perspectives from different data gathering techniques with the goal to validate the results, which has been advocated extensively in the HCI practice, see e.g., [91, 98]. To comply with this practice, we collected data from two questionnaires, datalogging and a semi-structural qualitative interview. Moreover, before this work on *NCCollab* started, we performed requirements gathering by collecting perspectives from self-identified practicing game developers through online forums.

We used the CSI survey [19] to compare the support of each interaction mode in terms of creativity. Our hypothesis was that the scores will be significantly different between the control condition and the two modes of collaboration, and also between the two collaborative modes. In reality, the results were found significant only in comparisons against the control condition, but not between the collaborative modes. CSI calculates the score by taking into consideration the weighted factor scores for collaboration, enjoyment, exploration, expressiveness, immersion, and results worth effort, as the dimensions of creativity support [19]. We found significant differences within the sample that we collected only for collaboration, which is likely the only influential factor contributing to the overall CSI score. This indicates that the participants felt all three modes provide comparable support for creativity, collaboration aside.

A self-developed questionnaire was created to solicit feedback on performance of each auxiliary technique that we implemented to support collaboration when used across the synchronous and asynchronous modes. Additionally, we used this

questionnaire to learn how participants felt about each mode in terms of confidence, efficiency, chance of future use, and overall. For confidence, significant differences were found between the control condition and the two collaborative modes. One interpretation could be that the participants felt more confident when working with a partner in contrast to doing solitary work. None of the participants had prior exposure to *NCCollab*, so they were still learning how to use the tool. We speculate that this could be due to the fact that the participants felt like they can rely on help from their partner in case they got stuck for whatever reason, which as a result, could cause less distress. Supporting this, it is worth noting that P8 and P10 suggested that the platform could be useful for teaching and learning due to the availability of live preview. This is in agreement with the vision behind solutions such a *Jimbo* [101]. *NCCollab* can be found useful in educational scenarios with its support of pair programming. For example, the instructors can give some tasks to the students and the students can engage in pair programming and the instructors can monitor their works in the live preview. This is especially useful for distance learning. However, as P8 has mentioned, this does not always need to apply in the situation where the skills of peer programmers differ. According to P8, collaborators with matching skill levels can benefit from this as well.

Unsurprisingly, participants rated vanilla *NodeCanvas* as an unlikely tool for future collaborative use. Interestingly, for chance of future use, significant difference was found between the control and the synchronous modes, but not the asynchronous mode. However, 58% of participants rated the asynchronous mode favorably compared to 50% who did the same for the synchronous mode. Furthermore, the synchronous mode was rated neutral 1.68 times more frequently than the asynchronous mode. These statistical findings aside, we think a broader perspective at the self-developed questionnaire findings suggest that both modes of collaboration have their applications.

In particular, P4 mentioned that the synchronous mode would be appropriate for quick prototyping.

During the study, we noted that BT history was utilized by 10 out of 12 participants. P12 was one of the participants who avoided using this feature and justified this due to the task having a small scale. P12 mentioned that instead they relied on their ability to easily keep track of the changes made to the BT in the mind. We speculate larger and more open-ended tasks will end up in more use of the feature.

Three participants wanted to have instant messaging with more advanced features such as voice chat. Other features that were suggested included referring nodes through messaging and wrapping the messages to fit into the screen as mentioned above. In the absence of these features, it is likely the users will resort to using business communication platforms such as *Discord* [26] and *Slack* [114]. We also noticed that all the participants either used conflict resolution or difference visualizations in the tree only after completing each task to check if their own BT fully matched with their partners' BT. 8 out of 12 used conflict resolution and four participants chose to use difference visualizations. However, none of the participants complained about difference visualizations, they were just preferred to use conflict resolution over difference visualizations. Overall, these findings may indicate that both of these features could be disruptive to the flow, resulting in participants using them after completing their task. This is also supported by the desire to have voice chat, as text messaging could be disruptive.

Similar to Nosek et al. [84], we logged how long it took the participants to complete the tasks. Both synchronous and asynchronous modes were faster than the control and the order effect was not significant. Our initial hypothesis was that the two collaborative modes would be different. In particular, we thought that in the

synchronous mode, since the task was done in tandem, the completion would be faster compared to the asynchronous mode where changes had to be integrated post-hoc at moments deemed appropriate by the participants. We also thought by working in tandem in real time, the participants would develop better strategies to avoid conflicts and coordinate their work better, which would expedite the completion. However, this is not what we found, since there was no significant difference between the collaborative modes. The unsurprising part of the findings was that both collaborative modes were faster than the control, which is typical to expect when the work is shared. This has been also known to be the case for code-based programming for decades [84]. It is worth acknowledging that a limitation in the form of a possibility may exist that the task itself could be a confounding factor since it varied across the levels of the independent variable.

P1 suggested adding a feature of locking a node so that the other users cannot make any changes to them. This is an interesting suggestion, however, arguably it would bring *NCCollab* closer to *SVN* [3], where users check out files and commit changes back to the server. While a file is checked out, it cannot be modified by other users. This is one of the reasons behind *SVN*'s waning popularity ever since the introduction of *Git* [39].

9.1 Limitations

In addition to the limitations that were stated above, it is also worth mentioning the following limitations of this work. First of all, to the best of our knowledge, our study is the first of its kind. As a result, there is no sample size to target from published related works as recommended by Mackenzie [72]. However, a recent study using a similar experimental design employed a sample of 10 participants [21]. In our study, the variance of the sample ahead of the study was not known to us. As a result, we could

not employ that information in the *a priori* power analysis. In determining the appropriate sample size we made an assumption that a large effect size of $\eta^2_p = 0.14$ or more is likely to be observed, not unlike in e.g., [21]. As a result, we targeted a sample size of 16 participants, which should have been sufficient given a minimal statistical power of $1- \beta = 0.8$ as recommended by Field [34]. In the end, due to the COVID-19 pandemic, the participation of 12 participants is all we were able to secure, which although is lower than needed, still happens to be the most common sample size within the CHI community [14]. Nonetheless, for CSI, the sample size was sufficient to detect the significance. However, the sample size may not have been enough for the analysis of other data that we collected and could be the reason behind multiple insignificant findings.

Another limitation is the fact that the participants have been recruited using convenience sampling, where their willingness to participate was prioritized over our need for random sampling from the population. As a result, generalizations of the results to the population will not be robust [98], p. 261. Also the participants were paid, which could potentially affect the outcomes.

At last, due to the COVID-19 pandemic, we performed the study online, which resulted in the variance between the specifications of the PCs which the participants used. Additionally, four participants used a second monitor. As a result, due to our inability to have a full control over the online experimental procedure and the hardware of the participants, there is a possibility that confounding variables may have been introduced.

Chapter 10
Conclusions & Future Work

10.1 NodeCanvas Alternatives

Game developers, like any other creative professionals, strive to explore design solutions in parallel, as they are looking for the optimal solution within a collection of similar options. The limitations of the design in current node-based solutions hinder this exploration, as users are forced to rewire and recreate nodes. As mentioned in the first part of this thesis, solutions to this problem have been proposed for other domains. In this work, we focused on game AI and behavior trees specifically, as this specialized domain affords for a unique opportunity for parallel, real-time and interactive exploration of multiple behaviors in the same game scene. Our work demonstrated that the interfaces for parallel game AI design are worth to consider in game development. In particular, we hypothesize that our diffing technique could prove to be superior to the text-based diffing for *Unreal Engine's Blueprints* [9].

In its current state, there were a few notable limitations with the workflow that we aim to improve in the future. Both *NC* and *NCAlt* do not support sufficient types of nodes to replace the scripts completely. Therefore, P3 and P15 did not find both *NC* and *NCAlt* that useful due to the absence of sufficient variations of nodes. Our focus was mainly on exploring alternatives and difference visualizations in this work. Therefore, in the future, more types of nodes can be integrated into *NCAlt* so that game developers can have more freedom when it comes to making AI. The theoretical number of alternatives that can be compared is limited by the number of distinct colors that can be used to designate each of them. For play test comparisons in the game view the limit is the maximum number of NPCs a user can effectively track simultaneously in the game scene, which was shown to be eight for slow moving objects [1]. For BT difference

visualizations, the number of alternatives that can be compared simultaneously is limited by the screen real estate with six BTs juxtaposed in a 2 × 3 grid likely being the practical limit on a typical monitor.

10.2 NodeCanvas Collaboration

Pair programming is common in software development. It, however, does not enjoy as much attention compared to visual programming. In recent times due to the COVID-19 pandemic, the demand for the support of pair programming for distance learning, be it text-based or node based, is what we expect to increase. Game development is a collective process where a variety of different professionals from different backgrounds cooperate. Although solutions for distributed collaboration as standalone applications exist, few are available that consolidate multiple collaborative features for specific tasks like game programming. Furthermore, none, to the best of our knowledge, exist for visual programming and game AI development in particular. As a result, *NCCollab* is our proposed solution which aims to shrink this gap.

Within the limitations of our second study, we found that *NCCollab* was received favorably by the participants, and that both modes of interactions can have their legitimate applications. It was suggested that the synchronous collaboration could be useful for quick prototyping, while the asynchronous collaboration could be appropriate for most other forms of collaboration. Some of the results we obtained were insignificant, which we speculate could be partially attributed to the small sample size that was employed in the study. As a result, further research is necessary to make stronger conclusions, which could be pursued in the future work. In our study, we focused on fixed tasks. We believe this is partially responsible for the fact that significant differences were not found in the analysis of the creativity support of *NCCollab*. In the future, longitudinal and open-ended tasks can be employed to study,

e.g., if an interaction effect between the mode of collaboration and creativity support exists.

References

[1] Alvarez, G.A. and Franconeri, S.L. 2007. How many objects can you track?: Evidence for a resource-limited attentive tracking mechanism. *Journal of Vision*. 7, 13 (Oct. 2007), 14–14. DOI:https://doi.org/10.1167/7.13.14.

[2] Andreas Schmeil, A., Martin Eppler, M. and Muraim Gubler, M. 2009. An Experimental Comparison of 3D Virtual Environments and Text Chat as Collaboration Tools. *Electronic Journal of Knowledge Management (EJKM)*. 7, 5 (Jan. 2009), 637–646.

[3] Apache Subversion: *https://subversion.apache.org/*. Accessed: 2020-11-01.

[4] Asana Project Management Software - Online Tools, Templates & App · Asana: *https://asana.com/uses/project-management*. Accessed: 2020-10-27.

[5] Behavior Designer - Behavior Trees for Everyone | Visual Scripting | Unity Asset Store: *https://assetstore.unity.com/packages/tools/visual-scripting/behavior-designer-behavior-trees-for-everyone-15277?gclid=Cj0KCQjw9b_4BRCMARIsADMUIypkySxldYUkWqI-HTUpPdYxjzc0Xy23V-YZG8x2T4mSASxIZr5LNmkaAvzIEALw_wcB*. Accessed: 2020-07-16.

[6] Behavior Trees: *https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/BehaviorTrees/index.html*. Accessed: 2020-08-04.

[7] Behavior Trees or Finite State Machines | Opsive: *https://opsive.com/support/documentation/behavior-designer/behavior-trees-or-finite-state-machines/*. Accessed: 2020-01-26.

[8] Berland, M., Davis, D. and Smith, C.P. 2015. AMOEBA: Designing for collaboration in computer science classrooms through live learning analytics. *International Journal of Computer-Supported Collaborative Learning*. 10, 4 (Dec. 2015), 425–447. DOI:https://doi.org/10.1007/s11412-015-9217-z.

[9] Blueprints Visual Scripting: *https://docs.unrealengine.com/en-us/Engine/Blueprints*. Accessed: 2018-12-22.

[10] Bolt: Visual Scripting for Unity: *https://ludiq.io/bolt*. Accessed: 2019-12-04.

[11] Bremm, S., von Landesberger, T., Heß, M., Schreck, T., Weil, P. and Hamacherk, K. 2011. Interactive visual comparison of multiple trees. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)* (Oct. 2011), 31–40.

[12] Brooke, J. 1996. SUS: A "Quick and Dirty" Usability Scale. *Usability evaluation in industry*. Taylor & Francis. 189–194.

[13] Build software better, together: *https://github.com*. Accessed: 2020-01-14.

[14] Caine, K. 2016. Local Standards for Sample Size at CHI. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, May 2016), 981–992.

[15] Card, S.K., Bongwon Sun, Pendleton, B.A., Heer, J. and Bodnar, J.W. 2006. Time Tree: Exploring Time Changing Hierarchies. *Visual Analytics Science and Technology, 2006 IEEE Symposium On* (Oct. 2006), 3–10.

[16] Carra, E. and Pellacini, F. 2019. SceneGit: a practical system for diffing and merging 3D environments. Association for Computing Machinery.

[17] Chen, H.-T., Wei, L.-Y. and Chang, C.-F. 2011. Nonlinear revision control for images. *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), 105:1–105:10.

[18] Cherry, E. and Latulipe, C. 2014. Quantifying the Creativity Support of Digital Tools Through the Creativity Support Index. *ACM Trans. Comput.-Hum. Interact.* 21, 4 (Jun. 2014), 21:1–21:25. DOI:https://doi.org/10.1145/2617588.

[19] Cherry, E. and Latulipe, C. 2014. Quantifying the Creativity Support of Digital Tools through the Creativity Support Index. *ACM Transactions on Computer-Human Interaction*. 21, 4 (Jun. 2014), 21:1–21:25. DOI:https://doi.org/10.1145/2617588.

[20] Chirigati, F., Freire, J., Koop, D. and Silva, C. 2013. VisTrails provenance traces for benchmarking. *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (Genoa, Italy, Mar. 2013), 323–324.

[21] Chu, E. and Zaman, L. 2021. Exploring alternatives with Unreal Engine's Blueprints Visual Scripting System. *Entertainment Computing*. 36, (Jan. 2021), 100388. DOI:https://doi.org/10.1016/j.entcom.2020.100388.

[22] Cristie, V. and Joyce, S. 2017. Capturing And Visualising Parametric Design Flow Through Interactive Web Versioning Snapshots. *IASS Annual Symposium 2017 "Interfaces - Architecture. Engineering. Science."* (Hamburg, Germany, Sep. 2017).

[23] Cristie, V. and Joyce, S. 2018. GHShot: 3D Design Versioning for Learning and Collaboration in the Web. (2018), 1–6.

[24] Dadgari, D. and Stuerzlinger, W. 2010. Novel User Interfaces for Diagram Versioning and Differencing. *British HCI* (2010).

[25] Dey, R. and Child, C. 2013. QL-BT: Enhancing behaviour tree design and implementation with Q-learning. *2013 IEEE Conference on Computational Inteligence in Games (CIG)* (Aug. 2013), 1–8.

[26] Discord - A New Way to Chat with Friends & Communities: *https://discord.com/channels/@me/517739346602754076*. Accessed: 2020-06-23.

[27] Doboš, J., Fan, C., Friston, S. and Wong, C. 2018. Screen space 3D diff: a fast and reliable method for real-time 3D differencing on the web. *Proceedings of the 23rd International ACM Conference on 3D Web Technology* (Poznań, Poland, Jun. 2018), 1–9.

[28] Doboš, J., Sons, K., Rubinstein, D., Slusallek, P. and Steed, A. 2013. XML3DRepo: a REST API for version controlled 3D assets on the web. *Proceedings of the 18th International Conference on 3D Web Technology* (San Sebastian, Spain, Jun. 2013), 47–55.

[29] Doboš, J. and Steed, A. 2012. 3D Diff: An Interactive Approach to Mesh Differencing and Conflict Resolution. *SIGGRAPH Asia 2012 Technical Briefs* (New York, NY, USA, 2012), 20:1–20:4.

[30] Doboš, J. and Steed, A. 2012. 3D Revision Control Framework. *Proceedings of the 17th International Conference on 3D Web Technology* (New York, NY, USA, 2012), 121–129.

[31] Dourish, P. and Bellotti, V. 1992. Awareness and coordination in shared workspaces. *Proceedings of the 1992 ACM conference on Computer-supported cooperative work* (New York, NY, USA, Dec. 1992), 107–114.

[32] Elkhaldi, M. and Woodbury, R. 2015. Interactive Design Exploration with Alt.Text. *International Journal of Architectural Computing*. 13, 2 (Jun. 2015), 103–122. DOI:https://doi.org/10.1260/1478-0771.13.2.103.

[33] Fan, H., Sun, C. and Shen, H. 2012. ATCoPE: any-time collaborative programming environment for seamless integration of real-time and non-real-time teamwork in software development. *Proceedings of the 17th ACM international conference on Supporting group work* (New York, NY, USA, Oct. 2012), 107–116.

[34] Field, A., Miles, J. and Field, Z. 2012. *Discovering Statistics Using R*. SAGE Publications Ltd.

[35] Firebase: *https://firebase.google.com/*. Accessed: 2020-10-20.

[36] FlowCanvas - Visual Scripting for Unity: *http://flowcanvas.paradoxnotion.com/*. Accessed: 2018-12-22.

[37] Forums - GameDev.net: *https://www.gamedev.net/forums/*. Accessed: 2020-10-25.
[38] Galy, E., Paxion, J. and Berthelon, C. 2018. Measuring mental workload with the NASA-TLX needs to examine each dimension rather than relying on the global score: an example with driving. *Ergonomics*. 61, 4 (Apr. 2018), 517–527. DOI:https://doi.org/10.1080/00140139.2017.1369583.
[39] Git: *https://git-scm.com/*. Accessed: 2018-12-22.
[40] Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C.D. and Roberts, J.C. 2011. Visual comparison for information visualization. *Information Visualization*. 10, 4 (Oct. 2011), 289–309. DOI:https://doi.org/10.1177/1473871611416549.
[41] Goldman, M., Greg D Little, G. and Robert C. Miller, R. Real-time collaborative coding in a web IDE | Proceedings of the 24th annual ACM symposium on User interface software and technology.
[42] Google Docs: *https://docs.google.com/document/u/0/*. Accessed: 2020-10-24.
[43] Google Meet: *https://meet.google.com/*. Accessed: 2020-10-27.
[44] Graham, M. and Kennedy, J. 2010. A survey of multiple tree visualisation. *Information Visualization*. 9, 4 (Dec. 2010), 235–252. DOI:http://dx.doi.org/10.1057/ivs.2009.29.
[45] Grasshopper^{TM}: *https://www.grasshopper3d.com/*. Accessed: 2018-12-22.
[46] Guenther, J.R. 2016. *Shiro - A language to represent alternatives*. Simon Fraser University.
[47] Guerra-Gómez, J.A., Buck-coleman, A., Pack, M.L., Plaisant, C. and Shneiderman, B. 2013. TreeVersity: Interactive Visualizations for Comparing Hierarchical Datasets. *Transportation Research Record (TRR), Journal of the Transportation Research Board (2013)*. (2013), 21.
[48] Guerra-Gómez, J.A., Buck-Coleman, A., Plaisant, C. and Shneiderman, B. 2011. TreeVersity: Comparing tree structures by topology and node's attributes differences. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)* (Oct. 2011), 275–276.
[49] Guerra-Gómez, J.A., Buck-coleman, A., Plaisant, C. and Shneiderman, B. 2012. TreeVersity: Visualizing Hierarchal Data for Value with Topology Changes. *Proceedings of the Digital Research Society 2012*. 2, (Jul. 2012), 640–653.
[50] Guerra-Gómez, J.A., Pack, M.L., Plaisant, C. and Shneiderman, B. 2013. Visualizing Change over Time Using Dynamic Hierarchies: TreeVersity2 and the StemView. *IEEE Transactions on Visualization and Computer Graphics*. 19, 12 (2013), 2566–2575. DOI:https://doi.org/10.1109/TVCG.2013.231.
[51] Hailpern, J., Hinterbichler, E., Leppert, C., Cook, D. and Bailey, B.P. 2007. TEAM STORM: Demonstrating an Interaction Model for Working with Multiple Ideas During Creative Group Work. *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition* (New York, NY, USA, 2007), 193–202.
[52] Hart, S.G. 2016. Nasa-Task Load Index (NASA-TLX); 20 Years Later: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. (Nov. 2016). DOI:https://doi.org/10.1177/154193120605000909.
[53] Hart, S.G. and Staveland, L.E. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in Psychology*. P.A. Hancock and N. Meshkati, eds. North-Holland. 139–183.
[54] Hartmann, B., Yu, L., Allison, A., Yang, Y. and Klemmer, S.R. 2008. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. *UIST 2008* (New York, NY, USA, 2008), 91–100.

[55] Hegde, R. and Dewan, P. 2008. Connecting Programming Environments to Support Ad-Hoc Collaboration. *2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (Sep. 2008), 178–187.
[56] Herman, I., Melançon, G., de Ruiter, M.M. and Delest, M. 1999. Latour — A Tree Visualisation System. *Graph Drawing* (Berlin, Heidelberg, 1999), 392–399.
[57] Hoek, A.V.D. 2004. Continuous coordination: a new paradigm for collaborative software engineering tools. *ICSE 2004* (2004).
[58] Iovino, M., Scukins, E., Styrud, J., Ögren, P. and Smith, C. 2020. A Survey of Behavior Trees in Robotics and AI. *arXiv:2005.05842 [cs]*. (May 2020).
[59] Jantke, K.P., Lunzer, A. and Fujima, J. 2005. Subjunctive Interfaces in Exploratory e-Learning. *Proceedings of the Third Biennial Conference on Professional Knowledge Management* (Berlin, Heidelberg, 2005), 176–188.
[60] Java | Oracle: *https://www.java.com/en/*. Accessed: 2020-10-28.
[61] Johansson, A. and Dell'Acqua, P. 2012. Comparing behavior trees and emotional behavior networks for NPCs. *2012 17th International Conference on Computer Games (CGAMES)* (Jul. 2012), 253–260.
[62] Kazi, R.H., Grossman, T., Cheong, H., Hashemi, A. and Fitzmaurice, G. 2017. DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design. *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2017), 401–414.
[63] Kirby, N. 2010. *Introduction to Game AI*. Cengage Learning PTR.
[64] Kolarić, S., Erhan, H. and Woodbury, R. 2017. CAMBRIA: Interacting with Multiple CAD Alternatives. *Computer-Aided Architectural Design. Future Trajectories* (Jul. 2017), 81–99.
[65] Kolarić, S., Woodbury, R. and Erhan, H. 2014. CAMBRIA: A Tool for Managing Multiple Design Alternatives. *Proceedings of the 2014 Companion Publication on Designing Interactive Systems* (New York, NY, USA, 2014), 81–84.
[66] Lunzer, A. and Hornbæk, K. 2006. An Enhanced Spreadsheet Supporting Calculation-Structure Variants, and Its Application to Web-Based Processing. *Federation over the Web* (2006), 143–158.
[67] Lunzer, A. and Hornbæk, K. 2006. RecipeSheet: Creating, Combining and Controlling Information Processors. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2006), 145–154.
[68] Lunzer, A. and Hornbæk, K. 2003. Side-by-side display and control of multiple scenarios: Subjunctive interfaces for exploring multi-attribute data. *Proceedings of OzCHI 2003*. (2003), 26–28.
[69] Lunzer, A. and Hornbæk, K. 2008. Subjunctive Interfaces: Extending Applications to Support Parallel Setup, Viewing and Control of Alternative Scenarios. *ACM TOCHI*. 14, 4 (Jan. 2008), 17:1–17:44. DOI:https://doi.org/10.1145/1314683.1314685.
[70] Lunzer, A. and Hornbæk, K. 2004. Usability Studies on a Visualisation for Parallel Display and Control of Alternative Scenarios. *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2004), 125–132.
[71] Maccormick, D. and Zaman, L. 2019. SuBViS: The Use of Subjunctive Visual Programming Environments for Exploring Alternatives in Game Development. *Foundations of Digital Games (FDG) 2019* (2019).
[72] MacKenzie, I.S. 2013. Chapter 5 - Designing HCI Experiments. *Human-computer Interaction*. I.S. MacKenzie, ed. Morgan Kaufmann. 157–189.

[73] Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J. and Shieber, S. 1997. Design galleries: a general approach to setting parameters for computer graphics and animation. *SIGGRAPH '97* (New York, NY, USA, 1997), 389–400.

[74] Microsoft OneDrive | Sign In or Sign up | Free Cloud Storage: *https://www.microsoft.com/en-ca/microsoft-365/onedrive/online-cloud-storage*. Accessed: 2020-10-24.

[75] Mohiuddin, A., Woodbury, R., Ashtari, N., Cichy, M. and Mueller, V. 2017. A Design Gallery System: Prototype and Evaluation. *ACADIA 2017: DISCIPLINES & DISRUPTION [Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) ISBN 978-0-692-96506-1] Cambridge, MA 2-4 November, 2017), pp. 414- 425* (2017).

[76] Moss, R. 7 examples of game AI that every developer should study.

[77] Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L. and Zhou, Y. 2003. TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility. *SIGGRAPH 2003*. 22, 3 (2003), 453–462. DOI:https://doi.org/10.1145/882262.882291.

[78] Namata, G.M., Staats, B., Getoor, L. and Shneiderman, B. 2007. A dual-view approach to interactive network visualization. *CIKM 2007* (Lisbon, Portugal, 2007), 939–942.

[79] Namco 1980. *Pac-Man*. Namco.

[80] Nancel, M. and Cockburn, A. 2014. Causality: a conceptual model of interaction history. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14* (Toronto, Ontario, Canada, 2014), 1777–1786.

[81] NodeCanvas - Asset Store: *https://assetstore.unity.com/packages/tools/visual-scripting/nodecanvas-14914*. Accessed: 2019-07-08.

[82] NodeCanvas - Behaviour Trees and State Machines for Unity Game Engine: *https://nodecanvas.paradoxnotion.com/*. Accessed: 2020-04-05.

[83] Noguchi, K., Gel, Y.R., Brunner, E. and Konietschke, F. 2012. nparLD: An R Software Package for the Nonparametric Analysis of Longitudinal Data in Factorial Experiments. *Journal of Statistical Software*. 50, 1 (Sep. 2012), 1–23. DOI:https://doi.org/10.18637/jss.v050.i12.

[84] Nosek, J.T. 1998. The case for collaborative programming. *Communications of the ACM*. 41, 3 (Mar. 1998), 105–108. DOI:https://doi.org/10.1145/272287.272333.

[85] Online Collaboration Tools for Modern Teams | Nulab: *https://nulab.com/*. Accessed: 2020-10-29.

[86] Online Diagram and Flowchart Software | Cacoo: *https://cacoo.com/*. Accessed: 2020-10-24.

[87] Online Project Management Software & Tools | Zoho Projects: *https://www.zoho.com/projects/*. Accessed: 2020-10-27.

[88] Open Broadcaster Software | OBS: *https://obsproject.com/*. Accessed: 2020-04-18.

[89] Parametric Modelling, Process, Advantages and Parametric Modelling Tools.: *https://www.designtechsys.com/articles/parametric-modelling*. Accessed: 2020-01-14.

[90] Pettersson, I., Lachner, F., Frison, A.-K., Riener, A. and Butz, A. 2018. A Bermuda Triangle?: A Review of Method Application and Triangulation in User Experience Evaluation. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), 461:1–461:16.

[91] Pettersson, I., Lachner, F., Frison, A.-K., Riener, A. and Butz, A. 2018. A Bermuda Triangle? A Review of Method Application and Triangulation in User Experience

Evaluation. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada, Apr. 2018), 1–16.
[92] Playmaker | Visual Scripting | Unity Asset Store: *https://assetstore.unity.com/packages/tools/visual-scripting/playmaker-368?gclid=Cj0KCQjw9b_4BRCMARIsADMUIyorQklboS6T1kXrXmFUnh4L61KJ4vXl1Pn6H9Q7U7g06Emhd0h8-_oaAqWSEALw_wcB*. Accessed: 2020-07-16.
[93] Programming and Web Development Help | DreamInCode.net: *https://www.dreamincode.net/*. Accessed: 2020-10-25.
[94] Project Management Software by ClickUp™: *https://clickup.com/teams/project-management*. Accessed: 2020-10-27.
[95] Rasmussen, J. Are Behavior Trees a Thing of the Past?
[96] Santoni, C., Salvati, G., Tibaldo, V. and Pellacini, F. 2018. LevelMerge: Collaborative Game Level Editing by Merging Labeled Graphs. *IEEE Computer Graphics and Applications*. 38, 4 (Jul. 2018), 71–83. DOI:https://doi.org/10.1109/MCG.2018.042731660.
[97] Scratch - Imagine, Program, Share: *https://scratch.mit.edu/*. Accessed: 2020-10-28.
[98] Sharp, H., Preece, J. and Rogers, Y. 2019. *Interaction Design: Beyond Human-Computer Interaction*. Wiley.
[99] Shireen, N., Erhan, H., Woodbury, R. and Wang, I. 2017. Making Sense of Design Space. *Computer-Aided Architectural Design. Future Trajectories* (Singapore, 2017), 191–211.
[100] Smith, B.N., Xu, A. and Bailey, B.P. 2010. Improving interaction models for generating and managing alternative ideas during early design work. *Graphics Interface 2010* (Toronto, Ont., Canada, Canada, 2010), 121–128.
[101] Soroush Ghorashi, S. and Jensen Carlos Jimbo | Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering.
[102] Tanimoto, S.L. 1990. VIVA: A visual language for image processing. *Journal of Visual Languages & Computing*. 1, 2 (Jun. 1990), 127–139. DOI:https://doi.org/10.1016/S1045-926X(05)80012-6.
[103] Team collaboration software | Backlog: *https://backlog.com/lp/team-collaboration/*. Accessed: 2020-10-27.
[104] Terry, M. and Mynatt, E.D. 2002. Recognizing creative needs in user interface design. *Creativity and Cognition 2002* (New York, NY, USA, 2002), 38–44.
[105] Terry, M., Mynatt, E.D., Nakakoji, K. and Yamamoto, Y. 2004. Variation in element and action: supporting simultaneous development of alternative solutions. *CHI 2004* (New York, NY, USA, 2004), 711–718.
[106] The Total Beginner's Guide to Game AI: *https://gamedev.net/tutorials/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942*. Accessed: 2020-10-16.
[107] TIGSource Forums - Index: *https://forums.tigsource.com/*. Accessed: 2020-10-25.
[108] Twiddla: *https://www.twiddla.com/*. Accessed: 2020-10-24.
[109] Unity: *https://unity.com/frontpage*. Accessed: 2019-12-10.
[110] Unity Forum: *https://forum.unity.com/*. Accessed: 2020-10-25.
[111] Unreal Engine: *https://www.unrealengine.com/*. Accessed: 2020-08-04.
[112] Valsamakis, Y., Savidis, A., Agapakis, E. and Katsarakis, A. 2020. Collaborative Visual Programming Workspace for Blockly. *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (Aug. 2020), 1–6.

[113] Video Conferencing, Web Conferencing, Webinars, Screen Sharing - Zoom: *https://zoom.us/*. Accessed: 2020-10-27.

[114] Welcome to your new HQ: *https://slack.com/intl/en-ca/*. Accessed: 2020-10-27.

[115] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F. and Wilson, J. 2020. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics*. 26, 1 (Jan. 2020), 56–65. DOI:https://doi.org/10.1109/TVCG.2019.2934619.

[116] Woodbury, R., Mohiuddin, A., Cichy, M. and Mueller, V. 2017. Interactive design galleries: A general approach to interacting with design alternatives. *Design Studies*. 52, (Sep. 2017), 40–72. DOI:https://doi.org/10.1016/j.destud.2017.05.001.

[117] Zacharis, N.Z. 2011. Measuring the Effects of Virtual Pair Programming in an Introductory Programming Java Course. *IEEE Transactions on Education*. 54, 1 (Feb. 2011), 168–170. DOI:https://doi.org/10.1109/TE.2010.2048328.

[118] Zaman, L., Kalra, A. and Stuerzlinger, W. 2011. The effect of animation, dual view, difference layers, and relative re-layout in hierarchical diagram differencing. *Graphics Interface 2011* (St. John's, Newfoundland, Canada, 2011), 183–190.

[119] Zaman, L., Neugebauer, C., Stuerzlinger, W. and Woodbury, R. 2018. GEM-NI+: Leveraging Difference Visualization and Multiple Displays for Supporting Multiple Complex Generative Design Alternatives. *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), LBW106:1–LBW106:6.

[120] Zaman, L., Stuerzlinger, W. and Neugebauer, C. 2017. MACE: A New Interface for Comparing and Editing of Multiple Alternative Documents for Generative Design. *Proceedings of the 2017 ACM Symposium on Document Engineering* (New York, NY, USA, 2017), 67–76.

[121] Zaman, L., Stuerzlinger, W., Neugebauer, C., Woodbury, R., Elkhaldi, M., Shireen, N. and Terry, M. 2015. GEM-NI: A System for Creating and Managing Alternatives In Generative Design. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2015), 1201–1210.

[122] Kodu - Visual Programming Language for creating Xbox games. *Microsoft Research*.

Appendices

Appendix A. *NCAlt* System Usability Scale (SUS) Survey

System Usability Scale

Instructions: For each of the following statements, mark one box that best describes your reactions to the website *today*.

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this website frequently.	☐	☐	☐	☐	☐
2.	I found this website unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought this website was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this website.	☐	☐	☐	☐	☐
5.	I found the various functions in this website were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this website.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this website very quickly.	☐	☐	☐	☐	☐
8.	I found this website very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this website.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this website.	☐	☐	☐	☐	☐

Appendix B. *NCAlt* Self Developed Pre-Study Questionnaires

User Study Questionnaires

* Required

Personal Information

1. **Age** *

2. **Gender** *
   Mark only one oval.
   - ◯ Male
   - ◯ Female
   - ◯ Prefer not to say
   - ◯ Other: _____

Pre-Study Questionnaire Part 1

3. **What does the term alternative mean to you?** *

4. **How long have you been using desktop/laptop computer (in years)?** *

5. **How many years of experience do you have with visual node-based programming languages? (Also known as diagrammatic or data-flow programming languages, which are based on the idea of "boxes and arrows")** *

6. **List all the node-based programming languages or tools that you have worked with and how many years of experience do you have with that particular tool? (e.g., Nodebox, vvvv, Max/MSP, Rhino3D, Node Canvas for Unity, Unreal Engine Blueprints, Flow Canvas for Unity, Bolt for Unity, Blender (Graphical Logic Editor or Material Nodes), etc.)** *

7. **How many years of experience do you have with block-based programming languages (Also known as Block-based coding, where all the programming language instructions are mainly represented as blocks. e.g., Scratch, Blockly, Snap!)?** *

8. **List all the block-based programming languages (e.g., Scratch,Blockly, Snap!, Alice etc.) that you have worked with. Indicate how many years of experience you have with each tool?** *

9. **How many years of experience do you have with flowcharts or similar diagrams in your line of work?** *

10. **Do you develop software outside of school assignments and projects (If yes, give details. If not applicable, type N/A)?** *

Pre-Study Questionnaire (Part 2)

11. **How many years of experience do you have with the Unity Engine?** *

12. **How many hours per week do you typically spend working with the Unity Engine?** *

13. **How many years of experience do you have with the Node Canvas in Unity?** *

14. **How many hours per week do you typically spend using the Node Canvas in Unity?** *

15. **How many years of experience do you have with the Flow Canvas in Unity?** *

16. **How many hours per week do you typically use the Flow Canvas in Unity?** *

17. **How many years of experience do you have with the Behaviour Designer in Unity?** *

18. **How many hours per week do you typically spend using the Behaviour Designer in Unity?** *

19. **On a scale of 1 to 7, rate your level of proficiency with C# in the context of Unity.** *

Mark only one oval per row.

	No Experience	Beginner	Below amateur	Amateur	Above amateur	Almost professional	Professional
proficiency with C# and Unity	◯	◯	◯	◯	◯	◯	◯

20. **How many years of experience do you have with the Unreal Engine?** *

21. **How many hours per week do you typically spend using the Unreal Engine?** *

22. **How many years of experience do you have with Unreal Engine's Behaviour tree?** *

23. **How many hours per week do you typically spend using Unreal Engine's Behaviour Tree?** *

24. **How many years of experience do you have with the Lumberyard Engine?** *

25. **How many hours per week do you typically spend using the Lumberyard Engine?** *

26. **How many years of experience do you have with the Modular Behavior Tree in Lumberyard Engine?** *

27. **How many hours per week do you typically spend using the Modular Behavior Tree in Lumberyard Engine?** *

28. **(OPTIONAL) If you have used other GUI based behaviour tree design tool list them below. Please describe how long you have been using each behaviour tree design tool and how many hours per week you typically spend using that particular tool?**

Appendix C. *NCAlt* Self Developed Post-Task Questionnaires

Post Task Questionnaires (Part 1)

30. **On a scale of 1 to 7, how would you rate the efficiency of the tool that you have just used for the task you were given?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Efficiency of the tool	◯	◯	◯	◯	◯	◯	◯

31. **On a scale of 1 to 7, how would you rate the ease of use of the tool that you have just used?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Ease of use	◯	◯	◯	◯	◯	◯	◯

32. **On a scale of 1 to 7, how would you rate the tool you have just used overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Rate overall	◯	◯	◯	◯	◯	◯	◯

33. **On a scale of 1 to 7, how often will you will use the tool in the future?** *
Mark only one oval per row.

	never	very rarely	rarely	occasionally	frequently	very frequently	always
Uses frequency	◯	◯	◯	◯	◯	◯	◯

34. **How many versions of Behaviour Tree alternatives were you able to manage effectively? Please explain.** *

35. **On a scale of 1 to 7, how would you rate creating multiple versions of a behaviour tree overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Multiple versions of behaviour tree							

36. **On a scale of 1 to 7, how would you rate merging nodes of a behaviour tree overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Merging nodes							

37. **On a scale of 1 to 7, how would you rate swapping nodes of a behaviour tree overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Swapping nodes							

38. **On a scale of 1 to 7, how would you rate comparing multiple behaviours in the game scene overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Comparing in game scene							

39. **On a scale of 1 to 7, how would you rate comparing multiple Behaviour Trees overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Comparing in behaviour tree							

40. **In your future game development project, how many Behaviour Tree alternatives will you likely be exploring for each game character (that requires behaviour) before selecting the desired option?** *

41. **What did you like about the tool that you have just used? List and rank the features you liked.** *

42. **Is there anything you disliked about the tool you have just used? List these issues (if any exist) and describe how these can be fixed or improve?**

43. **Are there any other comments you would like to add regarding the tool you have just used?**

Appendix D. *NCAlt* Self Developed Post-Task Questionnaires

Post Task Questionnaires (Part 2)

51. **On a scale of 1 to 7, how would you rate swapping nodes of a behaviour tree overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Swapping nodes	◯	◯	◯	◯	◯	◯	◯

52. **On a scale of 1 to 7, how would you rate comparing multiple behaviours in the game scene overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Comparing in game scene	◯	◯	◯	◯	◯	◯	◯

53. **On a scale of 1 to 7, how would you rate comparing multiple Behaviour Trees overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Comparing in behaviour tree	◯	◯	◯	◯	◯	◯	◯

54. **In your future game development project, how many Behaviour Tree alternatives will you likely be exploring for each game character (that requires behaviour) before selecting the desired option?** *

_____

55. **What did you like about the tool that you have just used? List and rank the features you liked.** *

_____

56. **Is there anything you disliked about the tool you have just used? List these issues (if any exist) and describe how these can be fixed or improve?**

_____

_____

44. **On a scale of 1 to 7, how would you rate the efficiency of the tool that you have just used for the task you were given?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Efficiency of the tool	◯	◯	◯	◯	◯	◯	◯

45. **On a scale of 1 to 7, how would you rate the ease of use of the tool that you have just used?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Ease of use	◯	◯	◯	◯	◯	◯	◯

46. **On a scale of 1 to 7, how would you rate the tool you have just used overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Rate overall	◯	◯	◯	◯	◯	◯	◯

47. **On a scale of 1 to 7, how often will you will use the tool in the future?** *
Mark only one oval per row.

	never	very rarely	rarely	occasionally	frequently	very frequently	always
Uses frequency	◯	◯	◯	◯	◯	◯	◯

48. **How many versions of Behaviour Tree alternatives were you able to manage effectively? Please explain.** *

_____

_____

_____

_____

_____

49. **On a scale of 1 to 7, how would you rate creating multiple versions of a behaviour tree overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Multiple versions of behaviour tree	◯	◯	◯	◯	◯	◯	◯

50. **On a scale of 1 to 7, how would you rate merging nodes of a behaviour tree overall?** *
Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Merging nodes	◯	◯	◯	◯	◯	◯	◯

57. **Are there any other comments you would like to add regarding the tool you have just used?**

Post Task Questionnaires (Part 3)

58. **What was your preferred tool to use during the tasks?** *

Mark only one oval.

- Node Canvas - Used for Task 1
- Node Canvas - Used for Task 2
- Other:

Appendix E. *NCAlt* Interview Questionnaires

NCAlt Interview Guide

CSI relevant:

Enjoyment:

- Do you believe you enjoyed using Node Canvasfor the task you were given? What about BTAlt? Please explain.

Exploration:

- Do you think Node Canvas influenced your to ability explore different ideas or outcomes? Was this effect positive or negative? Please explain. What about BTAlt?

Expressiveness:

- Do you think Node Canvas influenced your creative process? Was this effect negative or positive? Please explain how. What about BTAlt?

Immersion:

- Did Node Canvas distract you from your tasks? If so, please explain. What about BTAlt?
- On the other hand, do you believe Node Canvas helped you to get immersed into the task? Please explain. What about BTAlt?

Results Worth Effort:

- Did it feel like it was NOT worth the effort to use Node Canvas for the task? Please explain. What about BTAlt?
- Did it feel like it was worth the effort to use Node Canvas for the task? Please explain. What about BTAlt?

Ease of Use:

- Were there any features of Node Canvas that were difficult to use? Do you think this came in the way of the tasks you were given to complete? If so, please explain. What about BTAlt?
- Were there any features of Node Canvas that were easy to use? Do you think these features helped the task you were given to complete? Please explain. What about BTAlt?
- Do you see the intended purpose of Node Canvas ? How do you think could it be used regularly by other *Unity* developers? Please explain. What about BTAlt?

Creative outcomes

- Do you believe Node Canvas hindered your creative outcomes? Please explain. What about BTAlt?
- Do you believe Node Canvas resulted in you coming up with better creative outcomes? Does the support for alternatives has anything to do with this? Please explain. What about BTAlt?

Process:

- Do you believe the development process of behavior trees differs for Node Canvas compared to BTAlt? If it does, explain. If it does not please also explain.

Reflection:

- Do you believe Node Canvas affects how you think about exploring different behaviour trees before selecting the desired behaviour? Please explain. What about BTAlt?

Other:

- What did you like about the Node Canvas that you have just used? Please describe all the features you liked. Now please do the same for BTAlt.
- Is there anything you disliked about the tool you have just used? If so, please describe how these can be fixed or improved.
- Was it useful to change Behaviour trees in run-time in BTAlt or was it not useful? Please explain.
- Do you have any additional feedback?
- Do you have any suggestions for potential improvements to the UI/functionality?

Appendix F. *NCCollab* Pre-Development Requirement Gathering Survey

A Visual Scripting tool for Unity

I am planning to develop a visual scripting tool for my one of my research projects that supports collaboration and history tracking for game development. The following questions were designed to figure out which features would be valuable to have in such a tool. If you have time, please kindly provide your comments.

* Required

1. 1. Which tool or a resource do you use while developing games collaboratively? *

2. 2. Do you think that using collaboration features in a visual scripting tool (such as Unreal Blueprints, Amplify Shader Editor, Playmaker, Bolt, NodeCanvas) for game development would benefit you during game development? Why? If so, then how? *

3. 3. Tracking history allows access to previous states quickly. Do you think having this feature while developing games collaboratively using a visual scripting tool would be useful? Why? If so, then how? *

4. 4. Do you think the ability to perform collaborative game development in real-time using a visual scripting tool would be useful (e.g., like real-time collaborative editing in Google Docs)? If so, then how? Or do you think asynchronous collaborative visual scripting would be more useful? Why? *

Appendix G. *NCCollab* Pre-Study Questionnaires

Pre-Study Questionnaire (NCColab)

* Required

Personal Information

1. Age *

_____

2. Gender *

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Prefer not to say
- ◯ Other: _____

PC Specifications

For Windows:
1. Open Task Manager (Ctrl+Shift+Esc).
2. Click on the Performance tab.
3. The resulting window should show you the information you need including processor speed, memory, and graphics card information

For MacBook:
1.Click the Apple icon in the top left corner of your Mac.
2. This will bring up a drop-down menu. Pick the top option: About This Mac.
3. The resulting window should show you the information you need including processor speed, memory, and graphics card information

3. Please enter your CPU information. (e.g., Core i5, Core i7, Xeon) *

_____

4. Please enter your RAM information. (e.g., 8GB, 16 GB) *

5. Please enter your GPU information. (e.g., Intel HD Graphics 630) *

Pre-Study Questionnaire

6. Do you think that using collaboration features in a visual scripting tool (such as Unreal Blueprints, Playmaker, Bolt, NodeCanvas) for game development would benefit you during game development? If so, could you please explain how? *

7. How long have you been using desktop/laptop computer (in years)? *

8. On a scale of 1 to 5, rate your level of proficiency with visual scripting tools or systems. (e.g., Unreal Engine Blueprints, Flow Canvas for Unity, Bolt for Unity, Node Canvas for Unity, etc.) *

Mark only one oval per row.

	No Experience	Beginner	Average	Above average	Professional
Please rate	◯	◯	◯	◯	◯

9. How many years of experience do you have with the visual scripting tools? *

10. On a scale of 1 to 5, rate your level of proficiency with procedural, functional and object-oriented programming languages. (eg., Java, C, C++, C#, Python, R, Kotlin, Go, Swift, Javascript, etc.) *

Mark only one oval per row.

	No Experience	Beginner	Average	Above average	Professional
Please rate	◯	◯	◯	◯	◯

11. How many years of experience do you have with procedural, functional and object-oriented programming languages? *

12. How many years of experience do you have with the Unity Engine? *

13. How many years of experience do you have with Unreal Engine? *

14. List all the node-based programming languages or tools that you have worked with and how many years of experience you have with those particular tools? (e.g., Unreal Engine Blueprints, Flow Canvas for Unity, Bolt for Unity, Nodebox, vvvv, Max/MSP, Rhino3D, Node Canvas for Unity, Blender (Graphical Logic Editor or Material Nodes), etc.) *

15. List all the block-based programming languages (e.g., Scratch,Blockly, Snap!, Alice etc.) that you have worked with. Indicate how many years of experience you have with each of them? *

16. (OPTIONAL) If there is any other relevant experience you would like to mention, please write it below.

Appendix H. *NCCollab* Creativity Support Index (CSI) Survey

Please rate your agreement with the following statements:

I was satisfied with what I got out of the system or tool.

Highly Disagree **Highly Agree**

It was easy for me to explore many different ideas, options, designs, or outcomes, using this system or tool.

Highly Disagree **Highly Agree**

☐ N/A

The system or tool allowed other people to work with me easily.

Highly Disagree **Highly Agree**

I would be happy to use this system or tool on a regular basis.

Highly Disagree **Highly Agree**

I was able to be very creative while doing the activity inside this system or tool.

Highly Disagree **Highly Agree**

My attention was fully tuned to the activity, and I forgot about the system or tool that I was using.

Highly Disagree **Highly Agree**

Continue

Please rate your agreement with the following statements:

I enjoyed using this system or tool.

Highly Disagree **Highly Agree**

The system or tool was helpful in allowing me to track different ideas, outcomes, or possibilities.

Highly Disagree **Highly Agree**

What I was able to produce was worth the effort I had to exert to produce it.

Highly Disagree **Highly Agree**

The system or tool allowed me to be very expressive.

Highly Disagree **Highly Agree**

☐ N/A

It was really easy to share ideas and designs with other people inside this system or tool.

Highly Disagree **Highly Agree**

I became so absorbed in the activity that I forgot about the system or tool that I was using.

Highly Disagree **Highly Agree**

Continue

When doing this task, it's most important that I'm able to...

Explore many different ideas, outcomes, or possibilities ○ ○ Work with other people

1/15 Continue

When doing this task, it's most important that I'm able to...

Be creative and expressive ○ ○ Produce results that are worth the effort I put in

2/15 Continue

When doing this task, it's most important that I'm able to...

Enjoy using the system or tool ○ ○ Become immersed in the activity

3/15 Continue

When doing this task, it's most important that I'm able to...

Become immersed in the activity ○ ○ Produce results that are worth the effort I put in

4/15 Continue

When doing this task, it's most important that I'm able to...

Work with other people ○ ○ Enjoy using the system or tool

5/15 Continue

When doing this task, it's most important that I'm able to...

Produce results that are worth the effort I put in ○ ○ Explore many different ideas, outcomes, or possibilities

6/15 Continue

When doing this task, it's most important that I'm able to...

Be creative and expressive ○ ○ Become immersed in the activity

7/15 Continue

When doing this task, it's most important that I'm able to...

Work with other people ○ ○ Produce results that are worth the effort I put in

8/15 Continue

When doing this task, it's most important that I'm able to...

Be creative and expressive ○ ○ Enjoy using the system or tool

9/15 Continue

When doing this task, it's most important that I'm able to...

Explore many different ideas, outcomes, or possibilities ○ ○ Become immersed in the activity

10/15 Continue

When doing this task, it's most important that I'm able to...

Work with other people ○ ○ Be creative and expressive

11/15 Continue

When doing this task, it's most important that I'm able to...

Produce results that are worth the effort I put in ○ ○ Enjoy using the system or tool

12/15 Continue

When doing this task, it's most important that I'm able to...

Explore many different ideas, outcomes, or possibilities ○ ○ Be creative and expressive

13/15 Continue

When doing this task, it's most important that I'm able to...

Work with other people ○ ○ Become immersed in the activity

14/15 Continue

When doing this task, it's most important that I'm able to...

Explore many different ideas, outcomes, or possibilities ○ ○ Enjoy using the system or tool

15/15 Continue

Appendix I. *NCCollab* Post-Study Questionnaires (*Synchronous Collaboration*)

Post Task Questionnaires (NCColab Synchronized Collaboration)

* Required

1. Please enter your participant ID. *

2. On a scale of 1 to 5, how confident were you while using this variant of the tool for performing the task you were given? *

Mark only one oval per row.

	Not at all	Slightly	neutral	Very	Extremely
Please rate	◯	◯	◯	◯	◯

3. On a scale of 1 to 7, how would you rate the efficiency of the variant of the tool that you have just used for the task you were given? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

4. On a scale of 1 to 7, how would you rate the variant of the tool you have just used overall? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

5. On a scale of 1 to 7, if you had access to this variant of the tool, how often would you use it in your future game development projects? *

Mark only one oval per row.

	never	very rarely	rarely	occasionally	frequently	very frequently	always
Please rate	◯	◯	◯	◯	◯	◯	◯

6. If you had access to this variant of the tool in your future game development projects, would you prefer to work alone or collaboratively while using this tool for the given task? Please elaborate on your response. If you prefer collaboration, how many game developers will you likely involve in the collaboration? *

_____

_____

7. On a scale of 1 to 7, how would you rate conflict resolution during the collaboration overall? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

8. On a scale of 1 to 7, how would you rate resurrecting behavior trees from the past overall in this variant of the tool (BT history Window or Cltrl+z)? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

9. On a scale of 1 to 7, how would you rate the hierarchical tree representation in the BT History window in this variant of the tool? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

10. On a scale of 1 to 7, how would you rate the build in text messaging in this variant of the tool? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

Appendix J. *NCCollab* Post-Study Questionnaires (*Asynchronous Collaboration*)

Post Task Questionnaires (NCColab Asynchronous Collaboration)

* Required

1. Please enter your participant ID. *

2. On a scale of 1 to 5, how confident were you while using this variant of the tool for performing the task you were given? *

Mark only one oval per row.

	Not at all	Slightly	neutral	Very	Extremely
Please rate	◯	◯	◯	◯	◯

3. On a scale of 1 to 7, how would you rate the efficiency of the variant of the tool that you have just used for the task you were given? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

4. On a scale of 1 to 7, how would you rate the variant of the tool you have just used overall? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

5. On a scale of 1 to 7, if you had access to this variant of the tool, how often would you use it in your future game development projects? *

Mark only one oval per row.

	never	very rarely	rarely	occasionally	frequently	very frequently	always
Please rate	◯	◯	◯	◯	◯	◯	◯

6. If you had access to this variant of the tool in your future game development projects, would you prefer to work alone or collaboratively while using this tool for the given task? Please elaborate on your response. If you prefer collaboration, how many game developers will you likely involve in the collaboration? *

_____

_____

7. On a scale of 1 to 7, how would you rate conflict resolution during the collaboration overall? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

8. On a scale of 1 to 7, how would you rate resurrecting behavior trees from the past overall in this variant of the tool (BT history Window or Cltrl+z)? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

9. On a scale of 1 to 7, how would you rate the hierarchical tree representation in the BT History window in this variant of the tool? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

10. On a scale of 1 to 7, how would you rate the build in text messaging in this variant of the tool? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

Appendix J. *NCCollab* Post-Study Questionnaires (Unenhanced *NodeCanvas*)

Post Task Questionnaires (Unenhanced Node Canvas)

* Required

1. Please enter your participant ID. *

2. On a scale of 1 to 5, how confident were you while using this variant of the tool for performing the task you were given? *

Mark only one oval per row.

	Not at all	Slightly	neutral	Very	Extremely
Please rate	◯	◯	◯	◯	◯

3. On a scale of 1 to 7, how would you rate the efficiency of the variant of the tool that you have just used for the task you were given? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

4. On a scale of 1 to 7, how would you rate the variant of the tool you have just used overall? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

5. On a scale of 1 to 7, if you had access to this variant of the tool, how often would you use it in your future game development projects? *

Mark only one oval per row.

	never	very rarely	rarely	occasionally	frequently	very frequently	always
Please rate	◯	◯	◯	◯	◯	◯	◯

6. If you had access to this variant of the tool in your future game development projects, would you prefer to work alone or collaboratively while using this tool for the given task? Please elaborate on your response. If you prefer collaboration, how many game developers will you likely involve in the collaboration? *

_____

_____

7. On a scale of 1 to 7, how would you rate conflict resolution during the collaboration overall? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

8. On a scale of 1 to 7, how would you rate resurrecting behavior trees from the past overall in this variant of the tool (BT history Window or Cltrl+z)? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent
Please rate	◯	◯	◯	◯	◯	◯	◯

9. On a scale of 1 to 7, how would you rate the hierarchical tree representation in the BT History window in this variant of the tool? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

10. On a scale of 1 to 7, how would you rate the build in text messaging in this variant of the tool? *

Mark only one oval per row.

	very poor	poor	fair	neutral	good	very good	excellent	N/A
Please rate	◯	◯	◯	◯	◯	◯	◯	◯

Appendix K. *NCCollab* Completed Tasks Questionnaires

Completed Task Questionnaires (NCColab)

* Required

1. Please enter your participant ID. *

_____

2. What was your preferred variant of the tool to use during the tasks? *

Check all that apply.

- [ ] NCColab Synchronized Collaboration
- [ ] NCColab Asynchronized Collaboration
- [ ] Unenhanced Node Canvas
- [ ] None

Other: [ ] _____

3. Could you please elaborate on your chosen option above. *

_____

_____

Appendix L. *NCCollab* Interview Questionnaires

NCColab Interview Questionnaires

Purpose:

- Do you see the intended purpose of *NCColab* as an asynchronous and synchronous tool for collaborative behavior tree authoring? If so, could you please expand on that?

Ease of Use:

- Were there any features of the original Node Canvas that were easy to use for the given task? Do you think these features helped the task you were given to complete? Please explain.
- Were there any features of Synchronous *NCColab* that were easy to use? Do you think these features helped the task you were given to complete? Please explain.
- Were there any features of Asynchronous *NCColab* that were easy to use? Do you think these features helped the task you were given to complete? Please explain.
- Were there any features of Node Canvas, Synchronous *NCColab*, Asynchronous *NCColab* were difficult to use? Do you think this came in the way of the tasks you were given to complete? If so, please explain.

Process:

- Do you believe the development process of behavior trees differs for Node Canvas compared to Synchronous and Asynchronous *NCColab*? If it does, explain. If it does not please also explain.
- Do you think the development process of behavior trees differs between Synchronous and Asynchronous *NCColab*? If it does, explain. If it does not please also explain.

Types of Game:

- Do you think there is a genre of games where *NCColab* will work well? If so, please explain.
- Do you think there is a type of games where *NCColab* will NOT work very well? If so, please explain.

***NCColab* Features:**

- Do you think the ability to perform collaborative game development in real-time using a visual scripting tool would be useful? If so, then how? Or do you think asynchronous collaborative visual scripting would be more useful? When and how do you think someone would use this feature?
- Tracking history allows access to previous states quickly. Do you think having this feature while developing games collaboratively using a visual scripting tool would be useful? Why? If so, then how?

- Do you think, the hierarchical representation of the tree in *NCColab* helped you to visualize the tree in a small space? If so, please explain.
- Do you think, having the build in text messaging option in *NCColab* helped you to collaborate during the tasks? If so, please explain.
- Using live preview, you can instantly see what changes other developers are making to their canvases. Can you think of a use case where this feature would be really useful? If so, please explain.

Other:

- Which version of Node Canvas did like the most among unenhanced NodeCanvas, synchronous *NCColab* or asynchronous *NCColab*? Why?
  Synced.
- In your future game development, how many game developers will you likely involve in collaboration using *NCColab*? Please explain.
- If you could change certain things about each of the tools, what would they be?
- Is there anything else you disliked about the tool you have just used? If so, please describe how these can be fixed or improved.
- Is there anything that you wish to have in both NCAlt and *NCColab* for future work?
  Reducing the number of features.
- Do you have any additional feedback?