An investigation into the usability of level editor modalities in video games

By Alwyn Owen OWE14519159

A dissertation submitted in partial fulfilment of the requirements for the degree of

BSc (hons) Games Computing

School of Computer Science

University of Lincoln

2018

Acknowledgements

Thank you to my project supervisor Mr Cowe for his enthusiastic guidance and support throughout the creation of this artefact and paper.

I would also like to extend a massive thank you to all the participants who donated their time, without whom this research would have been impossible.

Contents

Abstract

Level editing software is an important part of game development, and can also be a catalyst for the growth of a community around a game if distributed. As such it is important to understand which level editing modalities users prefer most to ensure they can achieve their goals to the greatest effect. Not only this, but better level editors can enable more creativity and freedom for level designers, resulting in better levels. As such, this study aims to create a better understanding of level editing software by first understanding which level editing modality users prefer.

To achieve this, a 2D side scrolling motorbike game was created. It features both an integrated graphical level editor, and an external editor feature which allows for the use of a text editor to create levels. A convenience sample of participants were asked to create a level using both modalities, then filled in a questionnaire for each, recording their experiences. These results were then compared to determine which modalities users preferred.

The results show that all participants individually preferred the Graphical Level editor, however, not all participants disliked the Text based level editor. Some found it somewhat easy to use, while others found it very difficult, while all participants found the Graphical level editor easy to use. The results suggest that users prefer a Graphical Level Editor, however, a mixed approach that focuses mainly on Graphical editing with some text based features included may be better.

Areas where further research could be conducted are then discussed, with the aim of providing a broader and more concrete understanding of this topic.

Literature Review

The primary focus of this dissertation is on level editor design and implementation. The areas of research covered in this review include GUI (Graphical user interface) design, UX (user experience) design, and other game design research including level design research.

Metaphors in UI design

(Sylvester, 2013, 220-224) suggests that Metaphor is important for UI design in games. Suggesting that it can originate from many sources; whether that be real life objects, cultural archetypes and conventions, Game Clichés, or logical systems. However, Sylvester also warns that without the right "cultural priming" these metaphors may be "indecipherable", suggesting this is especially true for game clichés, since not all players will have played the same games and therefore may not be able to read or notices these visual clues. Furthermore, Sylvester states that each game must set up its own consistent "Metaphor Vocabulary", to ensure that players are able to reliably recognise the purpose of each metaphor type.

(Nardi and Zarmer, 1990, 11-16) argue that "The idea of using metaphors in the user interface is laudable" however they also state that metaphors are problematic for many reasons such as precision, practicality, and Prior knowledge. They state, "Metaphors are good at suggesting a general orientation, but not good at accurately encoding precise semantics". They also make mention of how metaphors can bring inaccurate baggage with them. Using the example of a trash can they note that some users may infer that they need to "set the trash can out for garbage pick-up on Friday mornings".

Nardi & Zarmer and Sylvester seem to agree on some aspects such as cultural priming but also disagree on other aspects. Sylvester suggests that Metaphors should be used since they intuitively provide a user with basic knowledge of something, without having to bore them with a tutorial. Sylvester also compares a games narrative and fiction as being one big metaphor, stating "imagine learning a complex video game if it were represented only by abstract shapes", suggesting that games require metaphors to be understandable to players. Whereas Nardi & Zarmer argue that they provide too much confusion and inaccuracy, which makes them an "inappropriate choice" for complex applications.

Some disagreement may source from the different context that both authors are writing from. Sylvester is writing specifically from a games point of view while Nardi & Zarmer are writing from a "complex scientific, engineering and business applications" point of view. However, user interfaces for games have many of the same requirements as complex software and in the case of a level designer require just as much user understanding of a system for the result to be engaging. Sylvester's best-known game "Rimworld" (Ludeon Studios, 2013) is a simulation management game, which allows the player to run a colony, which bears some level of similarity to complex business software. The prevalence of save icons resembling a 3.5" diskette in most modern software would suggest that metaphor can be used accurately. Even if the original meaning of the icon is lost on the user, it takes on a new meaning, and does more or less the same thing across all software.

Signals and Noise in UI design

(Sylvester, 2013, 225-227) suggests that all "information the player gets is part of a signal" (visual/audio/haptic/other information) which the player then processes and interprets to form an understanding. Sylvester states that "Noise is signal that fails to transmit meaningful information" citing "complex art and overcrowded signals" as two major causes of noise in video games.

Sylvester argues that good interface design reduces the amount of noise and maximises the amount of signal. This ensures that the player can comprehend what is happening on screen more quickly and with less effort. Sylvester quotes Mirrors Edge (EA DICE, 2008) as being a good example of a game that minimises noise through it's clean art style.

(Natoli, 2017) argues that noise in UI design is bad, stating "Good UI design is all about the signal". Natoli's definition has slightly different caveats but is mostly the same, one notable addition is noise in the form of "six splash screens in the app with no skip option", suggesting that noise is not only meaningless data in amongst useful data, but can also take a more extreme form of data that prohibits the user from getting to what they want to do, such as un-skippable tutorials.

Both Sylvester and Natoli warn about pursuing reduction of noise too far and the consequences this can have for visual hierarchy (or visual importance). Natoli warns that often designers remove visual cues that are useful to users to minimise noise, and that by doing so they may be increasing noise despite the reduction in on screen elements. For example, a designer may remove components that follow gestalt principles leaving only the raw information, despite there being less information on screen, it becomes more difficult for the user to locate what they are trying to find or do.

Visual Hierarchy in UI design

(Rodin, 2016) writes about the importance of visual hierarchy and describes many ways in which it can be achieved, citing multiple examples of it in practice in web design. (Sylvester, 2013, 227-230) suggests that this concept is even more important in video games, since it is possible for players to miss information. In the proposed motorbike game this is less the case since there is no narrative or objectives outside of drive to the end of the level. However, in the GUI level editor, visual Hierarchy will be important since the user will be presented with lots of data with multiple options per item. So, it is important to ensure that the most important items are given priority to ensure the editor is intuitive to use.

Flow

Flow is "a euphoric state of concentration and involvement" where "one becomes totally absorbed in an activity" and is "happy, motivated and cognitively efficient" (Johnson and Wiles, 2003). This makes it a very desirable state for a player to be in when using a level editor, since it will lead to higher productivity and a greater sense of accomplishment upon creating a level. Johnson and Wiles state that "An important precursor of flow is a match between one's perceived skills and the challenges associated with an activity", stating that an activity perceived to be too difficult can cause anxiety, and one too easy can lead to apathy. In a traditional game context this makes sense, however, within the context of creative software, being creative alone is difficult enough, therefore the software should not aim to make this task more challenging. This can make it more challenging for flow to be achieved through software design alone. For flow to be achieved it may be best for the software to stay out of the way and provide as direct a path as possible for the user to transfer their ideas to the level itself. It could also be beneficial for the software to facilitate happy little accidents, potentially leading to inspiration.

Johnson and Wiles go into detail describing what elements of Nielsen and Molich's guidelines for heuristic evaluation (Nielsen and Molich, 1990) overlap with flow. One example provided, is that games tend to provide minimal information, which helps to prevent the player from becoming distracted. They state that in some games "The entire screen is taken up with the action occurring and the user must deliberately request all other information". As such it would be beneficial to design a level editor using this principal of providing the minimum amount of information as possible while still allowing the player to gain access to further data and information should they request it. This should help to improve immersion in the level editor and reduce distractions.

Menus

In the GDC presentation Keeping Level Designers in the Zone Through Level Editor Design (Storm, 2016), Storm states around the 39-minute mark that searching through menus interrupts flow for level designers. This shows that ideally a level editor should feature as few menu's as possible for commonly accessed features. He goes on to say that

> "This kind of losing flow, or losing that bit of in the zone can have huge impacts. If the editor has so many menus that needs to be accessed all the time, it'll be almost impossible for the brain to reach that zone state"

> *"If I'm in flow, thinking of creative and awesome solutions for difficult problems in arts, scripting, design, anything… And suddenly I need to click through five tabs to that cool thing I thought about, my intention and relaxation is gone, not because I want it to be, but because I am a human being that is taken out of flow by clicking five tabs."*

By suggesting this, Storm shows that being taken out of flow can have large impacts for not only productivity (as a pure measure of time spent working), but can also lead to reduced quality of any levels created. This shows how important it is for level editing software to allow users to create levels without having to think about the software, since these inefficiencies can lead on to have huge knock on effects over time. He also states that developers of such tools should take on the mantra of "Just because the pipeline is functional, Doesn't mean the pipeline is done", since if a certain task is unintuitive or slow to complete, it will cause less creativity and slow level creators, especially if the task is repeated multiple times throughout the process of creating a level.

Storm also argues that less clicks is generally better in UI design. For example, if something can be done with one mouse drag, this is far better than requiring a few clicks + typing. Storm argues that while this only leads to marginal speedups in a single instance of an activity, that when the activity is repeated 10000 times, it equates to a significant difference. As such in many cases the additional development time to implement a feature is worthwhile. This equation improves further when you take into account his proposed spiral of good/bad user experience design.

Spiral of Good/Bad User Experience Design

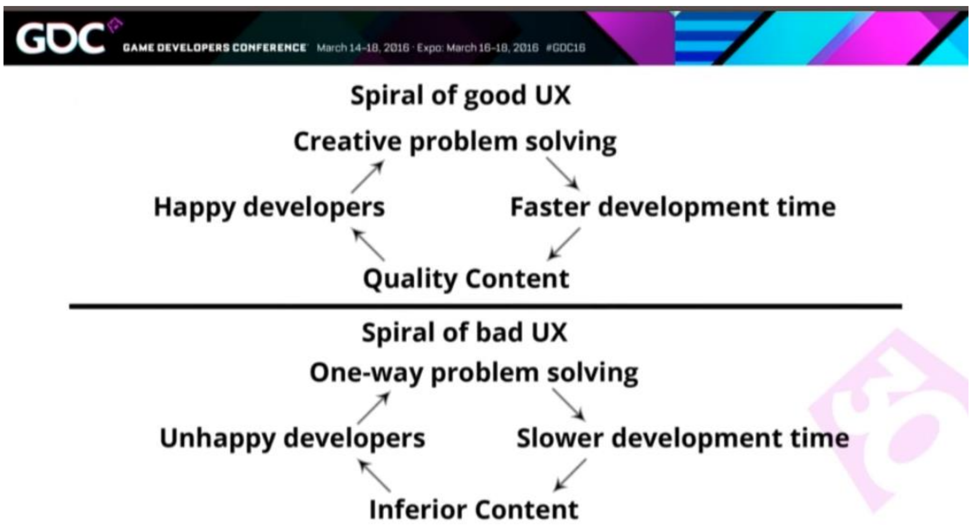


Figure 1: Storms Spiral of Good and Bad UX design (Storm, 2016)

Storm (Storm, 2016) also suggests during his GDC talk that bad design inside of tools doesn't have a one off or linear impact on the course of a project. He suggests that both good and bad design in an editor can have a spiralling effect on the quality of final product. This shows how further research into level editor design can be directly relevant to the industry. By better understanding which kinds of level editors people prefer, quality of products can be improved. Then additionally if an editor is designed to be shipped with a game, this could have a large impact on the games lifespan by kickstarting a modding/ content creation scene.

Playtesting efficiency

Storm (Storm, 2016) argues at around 50 minutes into his talk that requiring compilation in order to playtest a level leads to large inefficiencies. When discussing Hammer (Valve, 2004) Storm states that compiling a level can take "5 minutes, or up to an hour", this means that level designers may end up sitting around waiting only to find that they made a small mistake or that they want to make a slight change, resulting in having to repeat the process. Storm also argues that this is flow breaking, since the level designer would rather be able to test and then go directly back to editing. As such he comes to the conclusions that level editors that allow levels to be tested immediately are far better regarding play testing. As such it is important that the level editor created for this research allows the player to test with as little delay as possible.

Storm also suggest that it is beneficial to be able to keep the game and editor running at the same time. This was mostly regarding how some editors require the game to be rebooted whenever a level is ready to be tested. However, it would also be beneficial to be able to edit the level during gameplay too, since this reduces delays even further and allows for continuous testing while editing.

It is also suggested that allowing the player to be spawned anywhere in level can also further improve efficiency, by reducing the amount of times that a level needs to be replayed. However, it could be argued that this could lead to designers missing potential issues within levels, since they are likely to test them less often, and when they do, it may be without the full context of what has happened earlier.

Existing games

There are many similar games in this area. One example would be Line Rider (Čadež, 2006), it is a sandbox style game with no set objective, other than to explore the physics engine creatively and express one's self. It uses a minimalist UI with iconography representing all actions, each icon generally forms from some form of visual metaphor. This also aids with translation of the game to other languages, since there is near to no text to translate.

A higher production value example would be the Trials series of games (Red Lynx, 2014). Trials HD, Trials Evolution and Trials Fusion all featured comprehensive level editors allowing players to create and share levels. The level editors featured inside trials games are much more complex than those found inside a game like line rider. This is because Trials utilises a prop based system rather than a line based system, levels are created using individual items like railings (and explosives), rather than by drawing lines. Trials was predominantly designed for use with an xbox controller, this can be seen in its UI design where it makes use of Gestalt principles to create menus that are accessible without a pointing device, where the player can scroll between menu items and use a button to select them.

Methodology

Project Management

This project will have a single person working on it. Therefore, a project management methodology that is lightweight would be best suited. As such, Kanban was chosen.

Kanban is a simple methodology developed by Toyota. Its main benefits are reducing wasted time by limiting the number of ongoing tasks, and flexibility. It breaks projects down into individual tasks and assigns them into one of any number of pre-defined bins. This flexibility in the number of stages and work limits is useful for ensuring it remains relevant for a team of any size. Hence even for a team size of one it is useful for keeping track of tasks and ensuring that they get done. Kanban doesn't require a leader or dedicated person to control the process, this is also useful since it reduces the amount of overhead required to operate the process.

Many options were considered, but many were ruled out. The following are some of the methodologies that were considered and why they were not chosen.

Scrum

Scrum is a widely used agile project management methodology. It has many features that make it desirable for this type of project, burndown charts being one of the major unique features. Burndown charts allow a project's completion time to be estimated based upon current and past progress, as such project scope and time allocated can be adjusted to ensure that deadlines are met. However, in the context of project management, burndown charts are less useful if a single task such as software development is a multiple month period. If Scrum was used for both project management and software development this would once again be a useful feature.

One of the biggest disadvantages to scrum is that it is heavy. It is an involved process that usually requires a "Scrum leader" to orchestrate and control. When working as a team of one this is less practical, especially regarding daily meetings. Not only this but it also involves creating "User stories" for each task, this involves a lot of overhead, usually with the aim of providing programmers with exact tasks without requiring customer interaction. When there is only one team member, this level of abstraction has no benefit. However, it is still less heavy than PRINCE2.

PRINCE2

PRINCE2's biggest advantage is standardisation, it ensures that everyone knows each other's roles and is using common documents. However, for a single person project that benefit isn't relevant. As well as this, PRINCE2 is heavier than most other methods, it features ten "Roles" all of which have different responsibilities over different tasks within a project. This stems from PRINCE2's origins within large government organisations. It is stated in the PRINCE2 wiki (Prince2, 2017) "the Executive and the Project Management roles shouldn't be merged into one", for one person to fulfil both roles without merging them would be almost impossible. Therefore, PRINCE2 is not suitable for this project.

Software Development

This artefact will be created using object oriented principals. Therefore, a lightweight project management methodology that focuses on compartmentalised work is desirable. Once again, this made Kanban the ideal system to use.

Kanban for software development has all the same advantages as it does for project management. For example, individual tasks can be broken down into functions which can then be used in Kanban, this makes splitting larger tasks easier and helps to make them less intimidating.

The following are Software Development methodologies that were not chosen.

Extreme Programming

This methodology has many features that would be useful to this project, but not necessarily critical. For example, not programming things until they are required would help to ensure that the project reached minimum viable product as quickly as possible, this is helped further by the reduced initial planning. These benefits are most felt in projects where it is very likely requirements will shift over the course of development. However, for this project the requirements are mostly set in stone, with only smaller and less significant requirements that are not integral up for change. As such a lot of these features, while desirable, are not needed for this project, and the lack of other features such as planning are not ideal.

One of extreme programming's most prominent features is the use of pair programming. This is not possible with a team size of one. As many of Extreme Programming's principles rely on each other to make up for the shortfalls of its other principals, it is not ideal to pick and choose only some of its features. As such Extreme Programming is not suitable for this project.

Scrum

Scrum has the same advantages and disadvantages in the context of software development as it does for project management. As such it is too heavy for the scope of this project and the number of people who are working on it. However, again, its burndown charts would have been useful.

Toolsets

Windows

The Windows operating system (Microsoft, 2018) was used due to its standardised nature. It is installed on the vast majority of University lab computers and provides the best compatibility with Unity and Microsoft Office. All other comparable Operating Systems have much less market share.

Unity

Unity (Unity Technologies, 2018) is the game engine used for this artefact. Unity comes with built in rendering and physics which made it an attractive proposition over something more bare bones like SDL. Unlike Lumberyard which is built on CryEngine technology Unity fully supports 2d game development which was very important for this project. Unreal engine was a close contender to Unity, however Unreal is a heavier engine for small projects which may have made the end product more difficult to run on a portable machine. Unity is also quick to work with and features good documentation of its C# api's, whereas Unreal's C++ documentation is not so good.

Photoshop Elements 9

Photoshop Elements 9 (Adobe, 2010) was used to create the grass texture. This was done by creating a plain green texture and applying a filter.

Most image manipulation software could have created a similar texture and I didn't need any more advanced features in this case (see Aseprite below). I already own Photoshop Elements 9 therefore it would not have made sense for me to purchase Adobe Creative Cloud to get a more advanced software package, Adobe CC is also cost prohibitive. Krita (KDE, 2018) is a free software package that contains similar functionality to photoshop that could have also been used to create the grass texture.

Aseprite

Aseprite (Capello, 2018) was used to create the Motorbike texture and the icons for the GUI editor. Aseprite is an image manipulation program that is primarily focused toward pixel art. Photoshop Elements and Krita could have been used for this purpose however Aseprite has many features that make it better suited for this task. For example. Aseprite supports adding dual level pixel grids to the canvas to make it easier to work on small scales, this feature was especially useful for creating the pixel art for the game quickly and precisely. It also has a tickbox for choosing between precise pixel drawing and smoothed pixel drawing on the UI, this makes it very fast to switch between whatever is most appropriate for the current task, this meant that smooth lines like the motorbike shape could be drawn free hand with the smoothing algorithm, while the engine could be drawn pixel by pixel exactly. If animated sprites were to be added into the game at a later point, Aseprite supports onion skinning and animation with grid tile export options that are ideal for game engines.

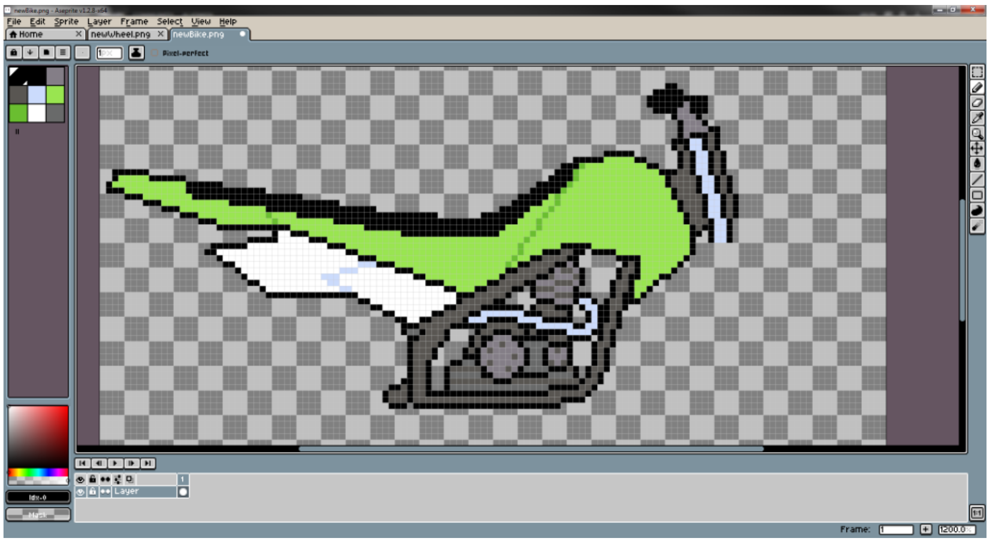


Figure 2: Aseprites dual layer pixel grid feature shown using the motorbike sprite

Visual Studio Enterprise 2015

Visual Studio (Microsoft, 2015) is the Industry standard IDE for programming in c#, its integration with Unity along with the ability to add plugins like ReSharper made it the obvious choice. MonoDevelop (Xamarin, 2018) would have been another option, it also integrates with Unity and provides the additional benefit of cross platform development. MonoDevelop does not support ReSharper, which made it a less attractive choice.

ReSharper

ReSharper (JetBrains, 2018) is a Visual Studio plugin that improves autocomplete and code highlighting functionalities within Visual Studio. This allows me to work faster with less bugs. Where there may be two ways of doing the same thing, ReSharper can change inefficient syntax for more efficient alternatives, which leads to better game performance.

Notepad++

Notepad++ (Ho, 2018) is used for creating levels with a text editor. It was chosen over notepad because of its features such as line numbering and formatting support. This ensures that any errors with level files can be found by referencing line numbers. Notepad++ is also a commonly used item of software, this should ensure that more participants feel at home when using the text based level editor. For participants who are unfamiliar with Notepad++, it functions similarly to other software which should allow them to use it more easily.

Microsoft Word

Microsoft Word (Microsoft, 2016) was used for creating the dissertation paper. It is the industry standard for document editing which makes it the obvious choice. It has an advanced and flexible feature set with an intuitive and quick user interface.

Microsoft Excel

Microsoft Excel (Microsoft, 2016) was used for creating a table of results. It is industry standard data manipulation software and has a wide feature set. It's ability to colour cells based upon the content within them allows it to provide more visual feedback that makes it easier to spot patterns.

Texts

Texts (Texts Software, 2017) is a minimalistic text editor. It is intentionally designed to have a minimal feature set with the aim of allowing the user to focus on writing rather than getting bogged down by formatting and styling. For this reason, it was used to keep text based notes on development progress and challenges faced, which I could then later use to assist with writing the dissertation paper. Notepad++ could have been used for this, but it does not support headings which would have made organisation more difficult. MS Word could have also been used, but it has a lot of unneeded features that would have distracted from the task at hand.

yEd

yEd (yWorks, 2017) was used to create the graphs in this dissertation paper. An alternative could have been to use MS Words built in graphing functionalities, however these can slow down document performance and are often difficult to use effectively. yEd allows for graphs to be created externally and imported as images.

Machine Environment

Since this project is research based, requiring participants to test the artefact, the only machine environment that it needs to run on is my laptop which is a Lenovo ThinkPad Yoga 12 (Lenovo, 2015). My Laptop is a portable device, this means it can be conveniently taken to any location where testing may be performed. It also includes all the equipment required for testing in a single box, which makes it easy to prepare and pack down after testing. The laptop is also capable of running Unity, this is useful for testing since it means that there is no need to compile the project into an exe file. However, the game should also run equally as well on any other system since it was created in Unity, which can produce standalone executables for a variety of operating systems.

A mouse was also used. This was because camera movement required the player to pan the camera using middle mouse click. On a trackpad or TrackPoint this was not possible. A mouse is also likely to be more familiar to many testers. Ideally the participants should not need to think about the control medium that they needed to use, so they had the option to use other input devices where they felt appropriate. The project also allowed participants to use the trackpad, TrackPoint, stylus, or touchscreen should they desire. The mouse used was a Logitech G500 (Logitech, 2009), this is because it works as well as any other mouse for the purposes of this project, therefore I saw no reason to purchase specific mouse hardware for user testing.

Research Methods

Quantitative or Qualitative?

This research will be using Likert scales on questionnaires. There will be two identical questionnaires for the participant to fill in, one for after each level editor modality. This will provide quantitative data which can be compared statistically. Which is useful for forming a conclusion as to which modality people prefer. This was most appropriate since it would less interpretation of results than sentiment analysis. For this reason, the questions asked were structured, to ensure that all responses could be compared like for like.

In their book "Research Methods" (Matthews and Ross, 2014) argue that thought needs to be given to "The validity of the data collection tool" when using structured data collection methods. As such a Likert scale (Likert, 1932) was used for data collection to ensure it was collected in a standardised method. Questions were designed to be as neutral and non-leading as possible while using an 'agree or disagree" response form. They were also designed to be short and concise, this is to ensure that participants can take in the whole question easily without struggling to comprehend what is being asked. No double negatives were used for the same reason.

If participants have any qualitative remarks that they would like to make, blank space is provided at the bottom of their response form. This will be used a secondary source of data to help justify the reasons they chose their responses. This was appropriate because some participants may feel the wish to justify their responses, equally it can provide further insight into common thoughts as well as potential issues that participants found with a certain method. This can help to identify if there were any shortcomings in the artefact.

How will data be gathered?

The research will use self-reported data from participants. This is appropriate for this type of study because it would be very difficult to measure what users prefer via other methods. For example, a participant's performance metrics could be tracked, but since all users will have different creative goals and abilities, data may be inconsistent between modalities, since participants may wish to put more effort into one level they make over the other depending on many factors outside of which editor is preferred. Alternatively, a participant could be asked to recreate a specific section of level, however, creativity is a large part of level creation, as such a test that ignores that aspect may come to inconclusive results.

The researcher will be present during all user testing, this ensures that participants have access to all the information they need during the testing. Care however must be taken to ensure this doesn't have an impact one results. Matthews and Ross state "This may affect the willingness of the respondent to participate or perhaps answer questions that are personal or sensitive.", for this research this I not a large problem since the data collected isn't sensitive. However, some users may feel uncomfortable reporting that they found something difficult to do, this may have an impact on the results.

Matthews and Ross also state "consideration has to be given to the impact the researcher as a social human being may have on the respondent", this is important to consider for this research as the researcher will be active during testing. As such it is important that the researcher will not provide any opinions about either modality and remain impartial, however, if a participant wants to know how to do something, or misunderstands how either editor functions, they will be given information on how to proceed. This may impact the results, however this research is ideally aimed at testing more than only novice users of a system, as such it is important to ensure that participants have access to any information they need.

It would be possible to observe participants using the software, however, this is very subjective and it is easy to miss things. If recordings are to be taken, this would also have implications for participant data security, it would be difficult to anonymize participants as such data gathered would likely need to be treated as not anonymous, which would make it more difficult to find willing participants and be outside the scope of this project. It is also possible that participants may behave differently when observed, especially with recording equipment and the knowledge that they would not be participating anonymously.

Another option that could have been used would be to use an exclusively long form answer response sheet, then create a word map for the output along with the use of sentiment analysis. This would provide a potentially stronger understanding of why users preferred a certain modality, but wouldn't have provided as strong of an objective understanding of how much they liked each.

What form should data take?

Since a Likert scale will be used, the data gathered will be ordinal. This is appropriate because it allows users to provide responses that can be gauged, for example, it would be possible to state that if a user gave the Text based editor a '3', and the GUI based editor a '5', it would be possible to infer that they felt the GUI editor was better in some regard.

How will results be represented?

Results will be represented in a table, with some additional graphs help visualise any patterns in the data. This is appropriate because the data is ordinal and quantitative, as such bar charts will be used. Comments made by participants will quoted in text. A word cloud could be made however there is not likely to be enough repeated text for it bring out anything relevant.

Validity

The order in which the two modalities were presented to participants switched every other participant to ensure that any bias that may exist towards the first or second method was equal for each modality.

A direct validation question was not asked; however, multiple questions can be combined in order to check for validity. Question 1, 7, & 8 stated:

"1. Creating a level was easy"

"7. The level editor hindered my ability"

"8. The level editor was confusing"

If a participant was to rate the above three questions positively it would suggest that they did not respond in a way that is accurate. Since it would be expected that if creating a level is easy, it wouldn't also be confusing and hinder your ability. With question 7 and 8 being 'agree = negative" questions and the rest being 'agree = positive' this should also be a way to check that participants have read the questions.

Standard deviations for results will be calculated to ensure that they are not spread randomly. As a result, low std. Distribution results would show that many participants agree on a certain question.

A two-tailed paired T-Test will also be used to test for significance.

Implementation

Research

Informed consent Evidence

Copies of all consent documentation can be found in the appendeces (items a,b,c).

All participants were over the age of 18.

Before any research was conducted, all participants were asked to read the "Participant Information Sheet" first, then asked to fill in the participant "Consent Form". Once the testing had finished all participants were given the "Participant Takeaway Form" which provides all information required to withdraw consent.

It was made clear in bold text on both the information sheet and takeaway form that participants are "free to withdraw from the research at any time prior to publication without prejudice". The consent form also made this clear by asking the participant to confirm that "I understand that my participation is voluntary and that I am free to withdraw at any time without giving a reason". Both above measures ensure that participants were aware of their right to withdraw consent.

The Participant information sheet made the purpose of the research and method by which the research will be conducted clear before asking the participant to provide consent. This ensures that the participants initial consent is well informed, and that they understand what will occur during testing.

Data collection information is also provided on the participant information sheet so that participants are aware of how their privacy will be handled. All data collected is anonymous, with all participants given a participant number that corresponds to their survey sheet.

The consent form contains tick boxes for the following statements

"I have read and understood the Participant Information Sheet"

This ensures that the Participant information sheet has been read and understood. As such the participant has been made fully aware of the research intent, methodology, there right to withdraw, and how their data will be handled.

"I have been given the opportunity to ask questions and have had them answered to my satisfaction"

This ensures that the participant has had the opportunity to receive further information or clarification on any points that they are unsure of. Thus, enduring their consent is fully informed.

"I agree to take part in this project"

This ensures that consent has been given and makes this clear to both parties.

"I understand that my participation is voluntary and that I am free to withdraw at any time without giving a reason"

This further reiterates that the participant is free to withdraw from the research should they wish to do so.

"I understand that my data will be kept and used anonymously"

This reiterates the key point of how data will be handled and asks the participant to agree to it.

The consent form design was influenced by existing designs from academic institutions. This was to ensure that it was fit for purpose and included everything that was required. The tick boxes were influenced by Newcastle Universities "Informed Consent Form Example" (Newcastle University, n.d.). The consent form and participant information sheet were also influenced by the University of Cambridge's "Consent forms and participant information sheets" guidance webpage (University of Cambridge, n.d.).

Procedure

Basic Overview

This diagram provides a basic overview of the process that each participant will engage with. Light blue boxes show how the participant interacts with the artefact. Dark blue boxes represent forms that the participant will fill in, the text for these dark blue boxes are above the main section of the diagram to reduce repetition of text and save horizontal space.

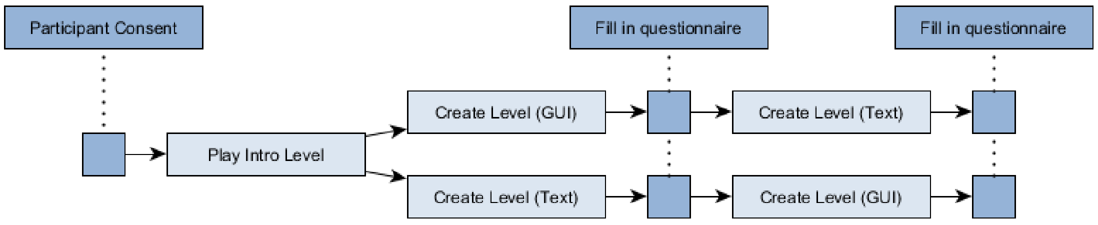


Figure 3: A graph to show the general flow of testing

Participant Consent

The participant is asked to read the Participant Information Sheet and fill in the consent form. If the participant does not consent to the testing, they may leave at any point.

Play Intro Level

The first thing the participant does, is play an Introductory level. This allows them to familiarise themselves with the game. During this introductory level the participant is taught the games controls and can see what a developed level for the game looks like, while also learning the physics and how the game plays.

Creating a Level

This is where the flow of the testing branched. Each participant used the opposite level editor to the previous first. This was to ensure that the order in which participants used each editor did not affect the results.

Each participant was asked to create a level of their own design, test it, and then fill in a questionnaire to record their opinions. This was done twice, once using each level editor. Both questionnaires were identical, the responses to these questionnaires will be compared to see which level editor modality is preferred amongst the group of tested participants.

Design

Physics

The physics system was originally designed to behave in a realistic manor, so the player would have control over the rider's centre of mass (their lower torso). This is like how riding a bike works in real life (GMBN, 2015). For example, to do a manual, the player would have to bring their hips down, and then back. This would be done by moving the analogue stick down, then pushing it left along the limit of the stick. Throttle and Braking would then be controlled by a separate axis such as the triggers.

File parsing and level loading

This was designed to with an algorithm intended to be flexible. It would look through the file to spot the start of a line, it would then iterate through until it found the start of a coordinate, the first number would be the 'x' value, then a comma, then a 'y' value, then a coordinate close. This coordinate section could be repeated as many times as needed, finally ending with a close line. Since the system would be checking against known syntax it should provide useful error messages with hints to suggest potential fixes for problems encountered. For example, if it reaches a close coordinate without finding a second coordinate, it may suggest the correct format to use for the user.

Booleans will be used as flags for the current position in the syntax. So, the loading system can tell what the user has missed out done wrong.

If any errors are made, only the affected areas of the level file should be abandoned to ensure that the rest of the level can still load as normal. While this may have the unfortunate side effect of allowing errors in completed levels, for debugging purposes it should make it easier for the user to spot where their mistakes may have been made.

JSON was considered for this purpose, however, it would have had more complex syntax for user to remember and would have resulted in larger file sizes for level files.

The level loading system wasn't planned down to small details, however, it was known that it should build geometry using trigonometry and display that in the scene.

External level editor

This was planned to allow the user to modify the level file outside of the game, then the game should detect the change and reload the level in game. This should only happen in a certain mode to prevent loss of data if the player is also working in the GUI based level editor. This was originally intended to occur upon a trigger.

Implementation

Physics

The physics section of implementation went faster than expected, it was intended to create custom physics code for the bike. However, Unity's built in RigidBody and joints system was fast to set up and did the same job. As such it was implemented instead. This had the benefit of allowing the inclusion of a physics modelled rider using connected bodies. This helps to provide additional feedback to the player, for example after a heavy landing the rider will move more violently. These physics required tweaking many times throughout development to ensure that they felt good.

The rider physics system went through multiple iterations.

1. Springs were attached to all body parts to hold them in place while still allowing the body to respond to impacts and jumps. The player controlled the length of the springs to move the riders body mass around the bike. Thus, allowing the player to control the balance.
2. Angle limits were added to joints in the second revision, this was to ensure that body hinges such as the riders elbow didn't bend the wrong way, since this would have been distracting and immersion breaking for players.

3. To provide the player with any reasonable amount of control of the riders balance it was found that the springs would need to be set up so stiffly that the rider physics started to break. The extra stiff springs caused vibrations which resulted in the rider moving constantly without player input, this had the visual effect of the rider continuously doing squats while riding.
4. Springs were removed from the riders physics system and replaced with hinges on each joint, with the idea being that forces could be applied to these hinges in order to keep the rider in a riding position. The players inputs could then control the riders position on the bike by influencing the position of these springs. The aim was to allow for more responsive movements while still allowing compliance in regard to absorbing hard landings.

After going through all these changes, it was found that the game was too difficult to be playable. This makes sense since it requires many years of practice to get good at riding a real motorbike to the level that this game would demand, so even with one less dimension to worry about it was still too difficult to control. As such the system was modified so that the player controlled the rotation of the bike directly rather than the rider's movements. This is a more arcade based approach since it doesn't mirror real world physics, however, it made the game playable. As such the rider was reverted back to using springs to stay in position, with hinge limits to prevent broken joints, the rider mass was then reduced so that it had little impact on the movement of the bike.

Unity's built in motor joint wasn't suitable for this project. The issue is surrounding how it is controlled within the code, it only has two variables which are 'speed' and 'torque'. Creating a system to control the speed and torque correctly like a real engine was attempted, but it was time consuming and didn't work. As such it was decided to write a custom solution by directly adding torque using the RigidBody, this solution was more elegant and required far less code.

File Parsing and Level Loading

File parsing went to plan. The error checking and syntax hint/suggestion systems functioned as expected. Code was then added so that if broken syntax was found, the line number would be reported as well as the contents of the broken line. This helps to ensure that debugging level files goes as smoothly as possible. As expected when there is an error in the level file the File Parsing and level loading system only throw away broken data allowing the rest of the level to be loaded correctly.

The level loading system was complex to implement due to the amount of geometry processing required. There was an unexpected issue with creating a collider for each line, since it required different formatting to Unity's rendering system, as such, a section of code was added to correctly format the data.

There is a known bug with the Level Loading geometry system. If a line goes back on itself (for example in a loop the loop) the geometry breaks. This is a fairly major bug, that was not fixed in time for participant testing. However, it should not affect the result since it effects both editing systems equally.

External Level Editor

This functionality went to plan, meaning the game checks the 'modified' date of the open level once every second. This means that when the player saves any modifications in an external level editor the level is reloaded automatically by the game. This was changed from the original plan since there is no trigger for file changes, polling for changes more often would have been less performant and polling less would have led to a noticable latency.

During testing of this feature, it was found that the game should pause, load the level, and then resume to ensure maximum physics stability. Since if a large level file were to be loaded it may take time to load, at which point the rider may have fallen through the floor otherwise.

This external level editor functionality is what the Text based editing system uses. Therefore any text editor of the users choice could in theory be used. As such it would even be possible for the user to create bespoke external level editing software should they feel the need.

GUI Level Editor

When trying to implement the puck system for moving line nodes, the Unity trigger system didn't seem to work. As such a custom mouse handling system was created that does not rely on triggers. This system allows for clicking to function and allows for dragging to function.

An optimisation was made for moving pucks that allows only the line that has been changed to be reloaded, this massively reduces the amount of work required compared to recalculating the entire level. This is important because the level is recalculated in real time as the player drags the level to provide visual feedback, so it need to be quick.

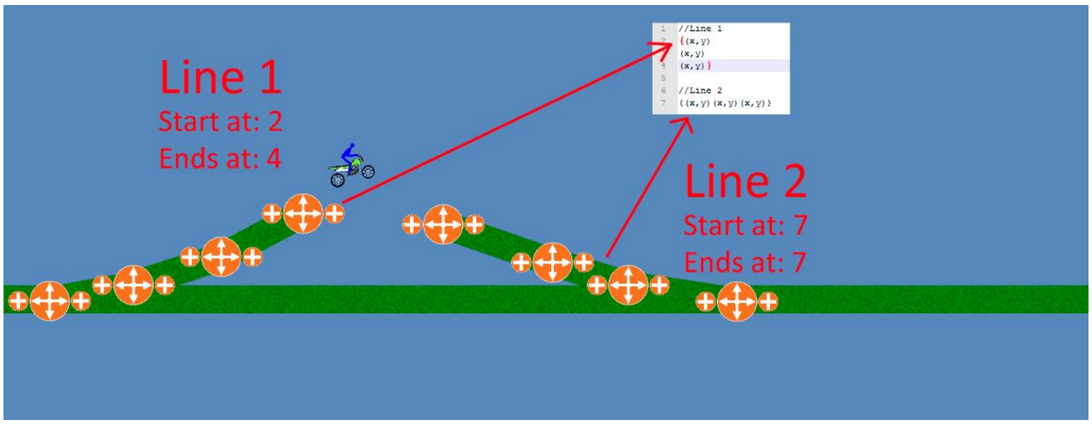
Deleting nodes went smoothly too, however it was found that it would be necessary to check how many nodes remained in a line. Since it is possible that a user could delete the last but one node, at which point there would have only been one node left, and it wouldn't have been possible to draw a line. As such, if a node is deleted and there is only one left, the entire line is deleted.

Writing the code to add new nodes into the middle of lines was troublesome. The code did not function as intended and some functions were mysteriously being called multiple times when they should have only been called once. After some attempts at bug fixing an infinite loop was created, while bad at the time this ultimately lead to the discover of the bug. The issue was that 'foreach' loops were being used to iterate over a line, however, since the line was being modified within the 'foreach' loop, the loop restarted its iteration leading to an infinite loop. This was fixed by switching to a 'for' loop.

File Saving

File saving was not originally intended to be a feature, since it was not needed for the research, therefore it was deemed to be outside of the projects scope. However, I had to create an introductory level for participants to play before creating their own, as such I would need to be able to save the level to a file.

Most of file saving was not too difficult, however, a large issue was encountered. If a player were to create a level in a text editor with comments and their own formatting, how can the software save without overwriting all their formatting and comments. Since the game allows line nodes to be defined over multiple lines of the save file binding each in game line to a text line was not enough. Instead all game lines were given the start and end line that they were defined on inside the text editor.

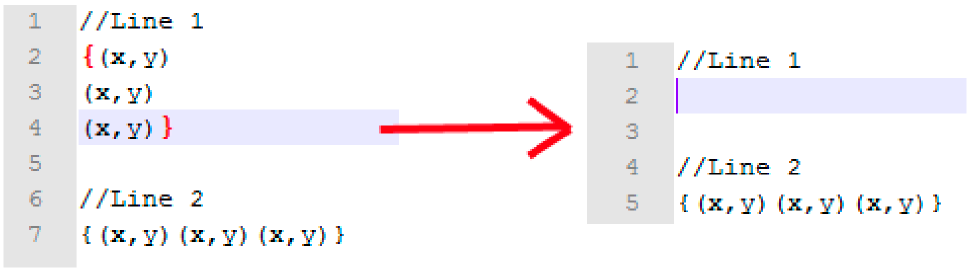


```
1 //Line 1
2 {(x,y)
3 (x,y)
4 (x,y)}
5
6 //Line 2
7 {(x,y)(x,y)(x,y)}
```

Figure 4: Image showing how two different lines could be represented in two separate ways inside the level file.

This allows the GUI level editor to only overwrite changes in the level file, as such all other formatting will be kept how it was originally.

This alone however was not enough to solve this issue. Since if a line was changed and overwritten, or a line were to be deleted, it would affect all other line number following it.



```
1 //Line 1
2 {(x,y)
3 (x,y)
4 (x,y)}
5
6 //Line 2
7 {(x,y)(x,y)(x,y)}
```

```
1 //Line 1
2
3
4 //Line 2
5 {(x,y)(x,y)(x,y)}
```

Figure 5: So, if "Line1" were to be deleted, "Line2" would no longer be on text row 7, it would have moved to text row 5.

As such, it was decided to add blank lines into the file to maintain the integrity of row numbers. A potentially better system would have been to update all line numbers or keep track of how they have changed while saving a level. However, adding blank space was the fastest solution to implement and functions perfectly within the scope of this project.

Testing

The game was tested throughout development with the completion of every feature. This was to ensure that each feature functioned as intended, while this testing strategy may seem like it should achieve 100% coverage, it does not, for new feature may break older seemingly unrelated features. As such, on top of this the game was tested after it was finished. This was done by going through the same process as participants and testing that the project functioned as intended when played. However, to properly test something, people other than the creator need to use it, since they are far more likely to spot problems or oversights.

As a result of this, once the game was completed a trial study of the research was done with 3 participants to ensure that the research procedures worked and that the game could fulfil its role within the research when used by real participants. In this case the program functioned as expected and the known geometry bug was not encountered by any of the 3 participants, this would suggest that the bug was not as bad as expected.

It would have been better to do more in depth testing with full path coverage, however, the scope of this project did not allow for that due to time constraints. External testing could have also had a positive impact, but the projects did not have the resources to conduct this form of testing.

This testing concluded that the game was fit for purpose.

During the trial run of the research, the paperwork was also used to check that it did not feature any issues. One issue was found in that the consent form had “name of participant” written in the researcher section. However, the heading still stated “Researcher” and that text didn’t cause any confusion, especially since it was not on an area of the form that participants were asked to fill in. Since it was only a minor detail, it was decided that it was not worthwhile to reprint the sheets to fix the error. During the actual research, no participants brought up this issue.

Evaluation

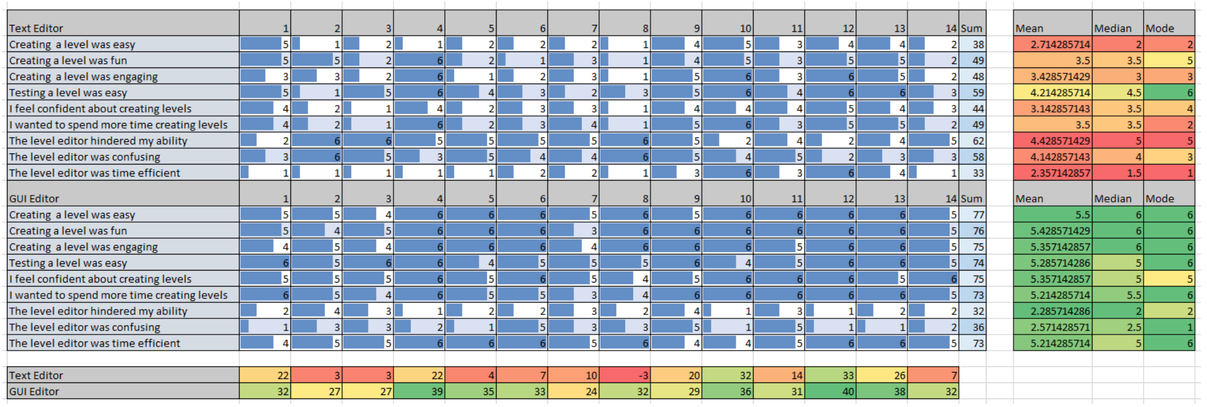
Participant Sampling

A convenience sample of participants were used. This includes friends, family and other known acquaintances. There are many reasons for this, most notably being time and resources available. There was limited time in which to complete the testing, which meant that it wasn’t easily feasible to assemble a truly representative and unknown group of participants. Due to the lack of resources it was not possible to correctly compensate people for their time and participation.

This may have influenced the results obtained. Since participants may have felt inclined to choose what they felt to be the “correct” answer rather than their true opinion.

The sample of participants covers a variety of people; however, it must be noted that many of them were other Computer Science and Games Computing students. This information was not collected as part of the research. However, there were participants ranging in age and gender, containing people both in their late teens and people over the age of 60. Roughly a quarter of participants are non-gamers.

Findings



Text Editor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum		Mean	Median	Mode
Creating a level was easy	5	1	2	1	2	2	2	1	4	5	3	4	4	2	38		2.714285714	2	2
Creating a level was fun	5	5	2	6	2	1	3	1	4	5	3	5	5	2	49		3.5	3.5	5
Creating a level was engaging	3	3	2	6	1	2	3	1	5	6	3	6	5	2	48		3.428571429	3	3
Testing a level was easy	5	1	5	6	4	3	2	3	5	6	4	6	6	3	59		4.214285714	4.5	6
I feel confident about creating levels	4	2	1	4	2	3	3	1	4	4	4	5	4	3	44		3.142857143	3.5	4
I wanted to spend more time creating levels	4	2	1	6	2	3	4	1	5	6	3	5	5	2	49		3.5	3.5	2
The level editor hindered my ability	2	6	6	5	5	5	5	6	5	2	4	2	4	5	62		4.428571429	5	5
The level editor was confusing	3	6	5	3	5	4	4	6	5	4	5	2	3	3	58		4.142857143	4	3
The level editor was time efficient	1	1	1	1	1	2	2	1	3	6	3	6	4	1	33		2.357142857	1.5	1
GUI Editor	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**Sum**		**Mean**	**Median**	**Mode**
Creating a level was easy	5	5	4	6	6	6	5	6	5	6	6	6	6	5	77		5.5	6	6
Creating a level was fun	5	4	5	6	6	6	3	6	6	6	6	6	6	5	76		5.428571429	6	6
Creating a level was engaging	4	5	4	6	6	6	4	6	6	6	5	6	6	5	75		5.357142857	6	6
Testing a level was easy	6	5	6	6	4	5	5	5	6	4	5	6	6	5	74		5.285714286	5	6
I feel confident about creating levels	5	5	5	6	5	6	5	4	5	6	6	6	5	6	75		5.357142857	5	5
I wanted to spend more time creating levels	6	5	4	6	5	5	3	4	6	6	6	6	6	5	73		5.214285714	5.5	6
The level editor hindered my ability	2	4	3	1	2	2	3	2	4	1	3	1	2	2	32		2.285714286	2	2
The level editor was confusing	1	3	3	2	1	5	3	3	5	1	5	1	1	2	36		2.571428571	2.5	1
The level editor was time efficient	4	5	5	6	6	6	5	6	4	4	5	6	6	5	73		5.214285714	5	6
Text Editor	22	3	3	22	4	7	10	-3	20	32	14	33	26	7					
GUI Editor	32	27	27	39	35	33	24	32	29	36	31	40	38	32					

Figure 6: A larger copy of this image can be found in appendix e.
NOTE: Scores were calculated by adding the values where a higher number is a positive response, then subtracting values where a higher score is a negative response. This is how participant 8 could give the text based editor an overall score of -3. The highest possible score is '40', the lowest possible score is '-5'.

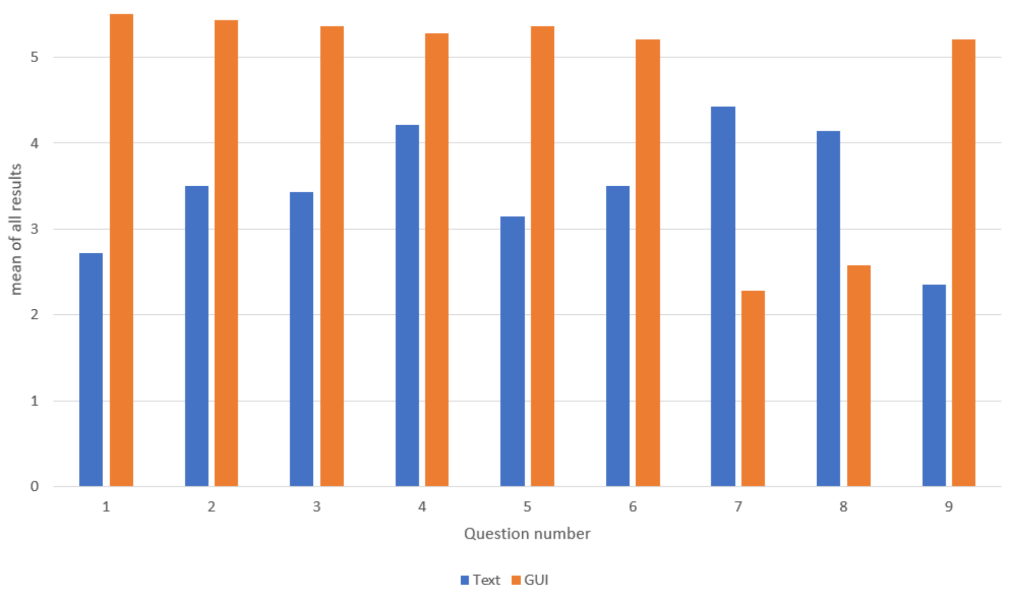


Figure 7: Graph showing mean values for questionnaire responses per question.
Graphs for Median and Mode can be found in appendix f.
NOTE: In Question 7 & 8, a higher value is a more negative response.

In general participants preferred the GUI level editor by a significant margin. Stating that it is both easier to use and that they enjoyed using it more. Participants found it slightly more difficult to test levels when creating levels with a text editor, however the difference was much smaller with identical Mode averages. Overall, Participants felt that the Text based level editor hindered their abilities and most users felt that it was not time efficient, however there was a minority that found it very time efficient.

Results were not identical across the board for all users. While the GUI editor scored consistently highly, the text editor was more polarising, most participants were not so keen on it, some others rated it well. However, not a single participant rated the text based editor higher than the GUI editor across all responses.

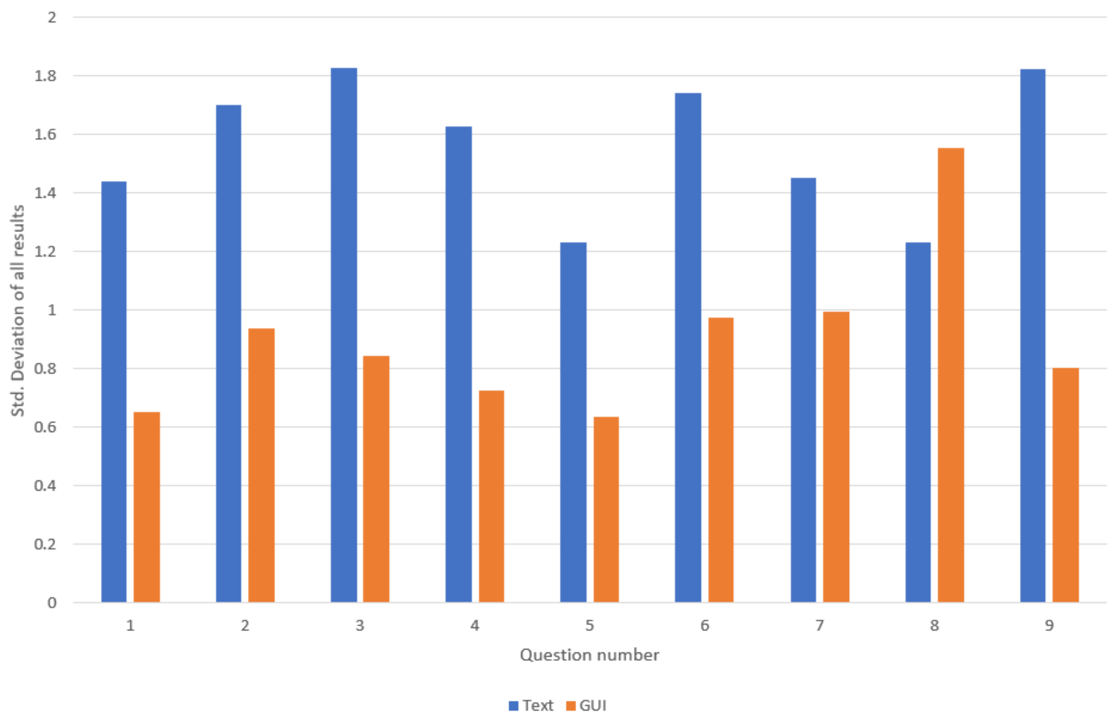


Figure 8: Graph showing standard deviation for each question on each questionnaire.

As shown in the graph above, the Text based editor had higher standard deviations than the GUI based editor in general. This shows that participants more consistently agreed with each other about the GUI editor, and had more diverse opinions about the Text based editor. Question 8 being the exception, showing that people disagreed over how confusing they found the GUI based editor more than with the Text based editor. The fact that 14 participants agreed to a Standard distribution within a single interval also lends itself to the significance of the results.

It is possible that training could have influenced results, however, participants spent similar amounts of time with each modality. The level of confusion participants felt when using the text based editor could be in part due to training. Therefore, it may be possible that with increased training participants may have felt more comfortable with the text based editor. On the other hand, training takes time, this extra time could be better used creating additional content or providing extra level creation training to improve the quality of levels produced. So, when compared like for like, these effects may negate each other.

Participants 10,12, and 13 tended to rate the text based editor more highly than other participants. There may be many reasons for this ranging from code literacy, to generally being mathematically inclined. One participant stated before creating a level with the text based editor that they felt they would be unable to use it, claiming that they are not good at maths and would not be able to understand it. This suggests that some people may find a text based editor to be daunting to approach.

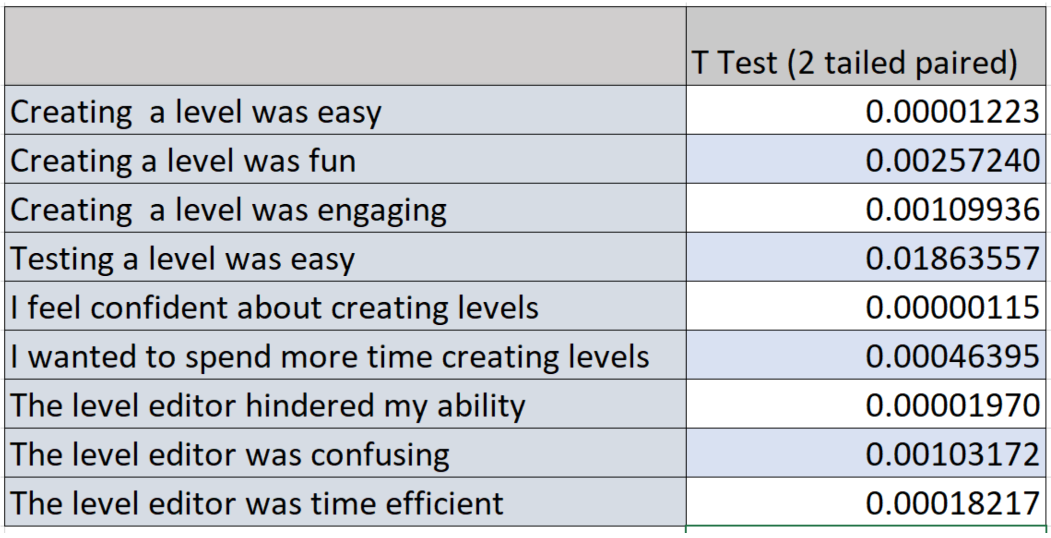
Participant 13 commented that the text based editor was “good for precise movements”, with participant 1 mirroring this sentiment by stating “preferred text for precision”. This suggests that they found some of the text based editor features compelling, even if their survey results show that on the whole they preferred the GUI editor. This may suggest that adding text based coordinates into the GUI editor may be appreciated by certain players who desire to achieve a more precise result. It may also suggest that they felt the GUI editor did not grant them the level of control that they may have desired.

Participant 13 also commented that it “would be nice to see coordinates in game”. Currently the game only displays updates to the level when the player saves the text file, this is due to file parsing issues that could occur. However, this comment shows that some players may prefer a continuously updating system that is smart enough to respond to incomplete syntax by highlighting coordinates on broken lines on screen. This behaviour would be somewhat like Microsoft Excel (Microsoft, 2016) where it highlights the cells that are being selected by syntax inside the formula bar.

Participant 9 wrote “Would have found it easier to just draw floor levels across just holding a key”. This shows that they would have preferred a drawing input system similar to how the pencil tool in Line Rider (Boštjan Čadež, 2006) works. This could be implemented by allowing the player to click and drag, detecting when the angle of the line being drawn exceeds a certain predetermined angle and creating a node each time this happens. Alternatively, a new node could be placed every ‘x’ distance to create a line.

Participant 3 wrote “No Undo” in reference to how the GUI editor does not have ‘undo’ functionality. The presence of ‘undo’ functionality may have helped users to feel more confident in the GUI level editor. This feature was present in the text editor since Notepad++ (Ho, 2018) supports this feature.

Significance of findings



	T Test (2 tailed paired)
Creating a level was easy	0.00001223
Creating a level was fun	0.00257240
Creating a level was engaging	0.00109936
Testing a level was easy	0.01863557
I feel confident about creating levels	0.00000115
I wanted to spend more time creating levels	0.00046395
The level editor hindered my ability	0.00001970
The level editor was confusing	0.00103172
The level editor was time efficient	0.00018217

Figure 9: Results of T-Test, showing P-Values

Doing T-Testing on the results shows that they are significant. “Testing a level was easy“, showed the least significance. This is interesting because it was the question that represented the least change within the artefact itself. For both GUI and Text based editors, the testing system remained the same, the only difference was how it was accessed, since it was instant in the GUI editor, and required the user to save and alt+tab with the text based editor. As such it makes sense that it is the question that had the least conclusive answer, even if the responses are statistically significant.

Overall the probability values given provide confidence in the results of this research.

Conclusion

Overall participants preferred the GUI editor, however there was a minority that still rated the text based editor well. This shows that if it is only possible to implement one solution, a GUI based solution ought to please the most people. However, if time/scope allows, it may be wise to offer the option of a text based editor or potentially add text based functionalities into the GUI editor as a form of nested complexity, this is an area that future research could expand upon.

More technically inclined participants liked the elements of the text based editor that allowed them improved precision and the general flexibility that a text based system allows for. This is shown further by the standard deviations for results. All users but one found the GUI editor to be more time efficient. All participants found that they felt the text based editor hindered their abilities more than the GUI based editor, which may be more important than other factors such as time efficiency because faster doesn’t necessarily mean a better result.

Efficiency is still important though, and participants felt that the GUI editor to be considerably more time efficient than the Text based editor. This could be due to warped perception of time as well as level of experience with each editor. However, there were a minority of participants who felt that the text editor was more efficient than most other participants gave it credit for. This would suggest that this factor depends more upon the user than the software itself. This is also reflected in it having the biggest difference in standard deviation between the two modalities.

The results are statistically significant and provide an interesting insight into the matter of level editors. These results are relatable to industry, which can use them to better inform decisions made regarding level editors, both for level designers and consumers.

These results do come with some caveats though. The sample size was smaller than ideal, and was a convenience sample. As such they may not accurately represent the target audience of any potential project that may aim to use these findings. Despite this, the sample was diverse in age range and moderately diverse in gender but still skewed male.

The artefact created was suitable for the research that needed to be conducted. All participants completed testing, not only this but many gave high scores to the GUI based editor. This would suggest that it was appropriate for the task at hand. As discussed, there are many improvements that could be made to it to further improve the experience of the Graphical editor, whether they would have made a significant difference to the results is not known, however it is somewhat unlikely considering the scores that participants gave it.

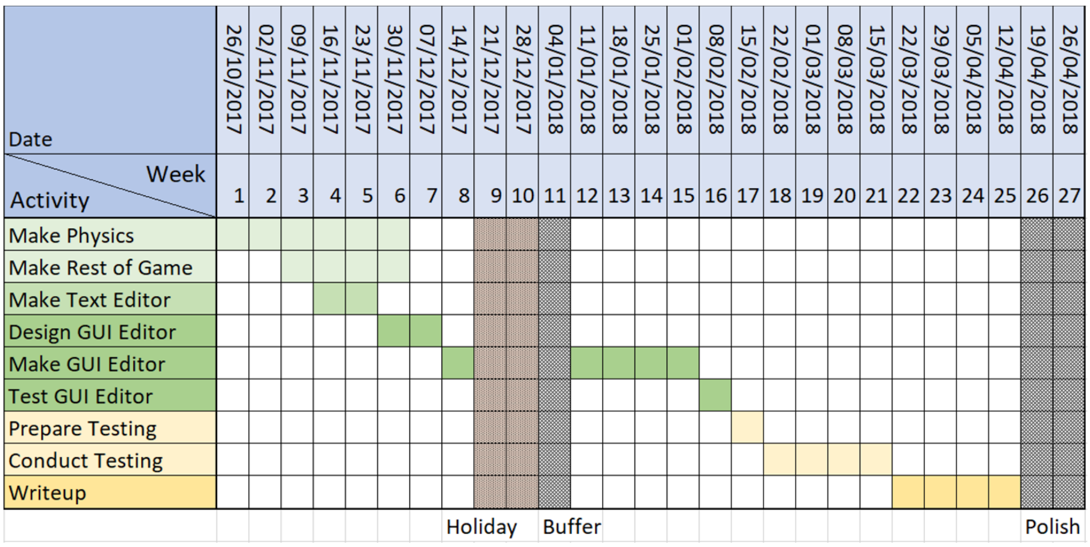
Reflective Analysis

Overall this project was successful within its scope; however, it was not perfect.

More participants with improved sampling could have made the results more useful to industry, by ensuring that they are representative of typical audiences. Being able to have multiple categories of participants, for example 10 participants per age grouping, could have led to more precise results that may show further trends on how to design for different groupings of people. However, within the time and financial constraints of this project this would not have been possible. As such it could be a good basis for follow up research of either a larger scope or of similar scope with a more targeted sample group. Since the artefact is already built at this point, it would be less time consuming to follow up on these aspects.

On the whole time was managed quite well. The artefact was completed close to schedule, and for most of the project the schedule was on time. However, towards the end of the project the schedule started to slip due to other assignments, this left less time than desired for participant testing and writeup. It may have been more optimal to make the artefacts deadlines earlier, to free up time for testing and writeup. Alternatively, more time overall would have allowed for more testing and a greater scope of project.

Original Gannt Chart



Date	26/10/2017	02/11/2017	09/11/2017	16/11/2017	23/11/2017	30/11/2017	07/12/2017	14/12/2017	21/12/2017	28/12/2017	04/01/2018	11/01/2018	18/01/2018	25/01/2018	01/02/2018	08/02/2018	15/02/2018	22/02/2018	01/03/2018	08/03/2018	15/03/2018	22/03/2018	29/03/2018	05/04/2018	12/04/2018	19/04/2018	26/04/2018
Week / Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Make Physics	🟩	🟩	🟩	🟩	🟩	🟩			▨	▨	▨															▨	▨
Make Rest of Game			🟩	🟩	🟩	🟩			▨	▨	▨															▨	▨
Make Text Editor				🟩	🟩				▨	▨	▨															▨	▨
Design GUI Editor						🟩	🟩		▨	▨	▨															▨	▨
Make GUI Editor								🟩	▨	▨	▨	🟩	🟩	🟩	🟩											▨	▨
Test GUI Editor									▨	▨	▨					🟩										▨	▨
Prepare Testing									▨	▨	▨						🟨									▨	▨
Conduct Testing									▨	▨	▨							🟨	🟨	🟨	🟨					▨	▨
Writeup									▨	▨	▨											🟨	🟨	🟨	🟨	▨	▨
								Holiday			Buffer															Polish	

Figure 10: The Gannt chart detailing the projects schedule

The original Gannt chart created for this project was mostly accurate and useful. Up until the Christmas holiday the project was on schedule, however after the Christmas holiday the project was one week behind, however this was not a major issue. The project remained one week behind until user testing began, this proved more time consuming than expected, which was compounded by deadlines for other modules occurring at that time. This meant a lot of the focus was shifted away from the project and towards other assignments. As a result, the project was pushed back by another two to three weeks leading to a larger than planned amount of work in the last three weeks.

From this I have learnt, that when creating schedules, they need to take into account other projects and their requirements rather than only requirements of the project being scheduled. And, that for a schedule to be effective, other projects should also have a schedule to mitigate the possibilities of crunch from one project effecting another project. As such I have learnt that schedules can be an effective way of planning time and ensuring that things stay on track, as the above one achieved that goal for most of its duration.

I also learnt about the time investment required for getting participants to participate. This was underestimated in the project, and is a consideration that I will take forward for future projects. Since while testing may only take 10-15 mins per participant, finding willing participants takes far longer. The number of participants in this research was less than intended because of lack of time towards the end, as already discussed, this could have been improved by better planning.

This project reinforced my appreciation of Object Oriented programming. Many of the additional unplanned features such as file saving could have been very difficult to add into the project at a later point if pure functional programming had been used.

The process and significance of doing a literature review also became important. Through finding out about the work of others, it is possible to drastically improve the relevance and quality of your own work.

If I were to do the research segment of this project again, I would want to add more qualitative questions onto the survey. This would allow for better explanations to be given for the patterns shown in the data. While this falls within the goal of creating a greater understanding of level editors, it may have proven to be feature creep if it was actually implemented.

Additional artefact features

There are many additional features that could be added to the GUI editor that may improve its usability or functionality.

If more time was available, it would be nice to implement the line drawing feature suggested by participant 9 as well as allowing coordinates to be changed via text inside the GUI editor. Further research could then be done by comparing all three level editing modalities using a similar test methodology. This would help to clarify whether a mixed approach to level editors is preferred by players.

It would also be nice to add undo functionality into the GUI editor, since it’s lack of presence puts it at a slight disadvantage. However, this feature may be of large scope since undo functionality is not to dissimilar to time travel, and can therefore be the source of many bugs and issues within software.

Snapping may be useful for some users in the GUI editor. Allowing lines to be snapped to other lines to join them together as well as a grid snap could both be useful features to improve the flow when creating levels. This was a feature that Storm (Storm, 2016) suggested is useful in his comparison between different level editing packages. While its importance is lessened in a 2d environment compared to the 3d editors that he compared, it should still improve quality of life.

Multiselect would also provide a large quality of life improvement to users, by allowing them to select multiple nodes to manipulate at once, potentially all nodes in one line for example. This would allow levels to be constructed in a more modular manor with much greater control of things after they have been built.

References

Adobe. (2010) Photoshop Elements 9 [software]. Available from https://www.adobe.com/uk/products/photoshop-elements.html [accessed 25 April 2018].

Čadež, B. (2006) Line Rider [game]. Slovenia: Čadež, B. Available from https://www.linerider.com/ [accessed 20 April 2018].

Capello, D. (2018) Aseprite [software]. Available from https://www.aseprite.org/ [accessed 25 April 2018].

EA DICE (2008) Mirror's Edge [game]. California: Electronic Arts. Available from https://store.steampowered.com/app/17410/Mirrors_Edge/ [accessed 15 April 2018].

GMBN. (2015) How To Manual Like A Pro – MTB Skills [video]. Available from https://www.youtube.com/watch?v=NkWnV4RDzkU [accessed 25 April 2018].

Ho, D. (2018) Notepad++ [Software]. France: Ho, D. Available from https://notepad-plus-plus.org/ [accessed 20 April 2018].

JetBrains. (2018) ReSharper [plugin]. Available from https://www.jetbrains.com/resharper/ [accessed 25 April 2018].

Johnson, D. Wiles, J. (2003) Effective affective user interface design in games. Ergonomics, 46(13-14) 1332-1345.

KDE. (2018) Krita [software]. Available from https://krita.org/en/ [Accessed 20 April 2018).

Lenovo. (2015) Lenovo ThinkPad Yoga 12 [laptop]. Available from https://www3.lenovo.com/us/en/laptops/thinkpad/yoga-series/yoga-12/ [accessed 25 April 2018].

Likert, R. (1932) A technique for the measurement of attitudes. Research Methods. Harlow: Longman. Available from http://psycnet.apa.org/record/1933-01885-001 [accessed 23 April 2018].

Logitech. (2009) G500 [mouse].

Ludeon Studios (2013) Rimworld [game]. Montreal: Ludeon Studios. Available from https://rimworldgame.com [accessed 15 April 2018].

Matthews, b. Ross, L. (2014) Research Methods: A Practical Guide for the Social Sciences. London: Pearson. Available from http://catalogue.pearsoned.co.uk/educator/product/Research-Methods-A-Practical-Guide-for-the-Social-Sciences/9781405858502.page [accessed 23 April 2018].

Microsoft. (2016) Excel [Software]. Redmond: Microsoft. Available from https://products.office.com/en-gb/excel [accessed 20 April 2018].

Microsoft. (2015) Visual Studio Enterprise [software]. Available from https://www.visualstudio.com/vs/older-downloads/ [accessed 25 April 2018].

Microsoft. (2018) Windows [operating system]. Available from https://www.microsoft.com/en-gb/windows [accessed 20 April 2018].

Microsoft. (2016) Word [Software]. Redmond: Microsoft. Available from https://products.office.com/en-gb/word [accessed 20 April 2018].

Nardi, B. and Zarmer, C. (1990) Beyond Models and Metaphors: Visual Formalisms in User Interface Design. Hewlett Packard.

Natoli, J. (2017) Signal vs. Noise: Removing Visual Clutter in the UI. Available from http://www.givegoodux.com/signal-vs-noise-cleaning-up-visual-clutter-in-ui-design/ [accessed 15 April 2018].

Newcastle University. (n.d.) Informed Consent Form Example. Available from http://www.ncl.ac.uk/res/research/ethics_governance/ethics/toolkit/consent/consent_form_example.doc [accessed 20 March 2018].

Nielsen, J. Molich, R. (1990) Heuristic evaluation of user interfaces. In: CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Seattle, Washington, 1-5 April. New York, USA: ACM, 249-256. Available from https://dl.acm.org/citation.cfm?id=97281 [accessed 19 April 2018].

Prince2. (2017) Roles and responsibilities. Available from http://prince2.wiki/Roles_and_responsibilities [accessed 17 April 2018]

Red Lynx. (2014) Trials Fusion [game]. Montreuil: Ubisoft. Available from https://www.ubisoft.com/en-us/game/trials-fusion/ [accessed 23 April 2018].

Rodin, R. (2016) A graphic Designer's Guide to Visual Hierarchy. Available from https://zevendesign.com/designers-guide-to-visual-hierarchy/ [accessed 16 April 2018].

Storm, R. (2016) Keeping Level Designers in the Zone Through Level Editor Design. In: GDC 2016, 14-18 March, California, United States. Available from https://www.youtube.com/watch?v=brByJ5EVBn4 [accessed 21 April 2018].

Sylvester, T. (2013) Designing Games: A Guide to Engineering Experiences. California: O'REILLY.

Texts Software. (2017) Texts [software]. Available from http://www.texts.io/ [accessed 25 April 2018].

Unity Technologies. (2018) Unity [software]. Available from https://unity3d.com/ [accessed 25 April].

University of Cambridge. (n.d.) Consent forms and participant information sheets. Available from https://www.research-integrity.admin.cam.ac.uk/research-ethics/ethics-application-guidance/consent-forms-and-participant-information-sheets [accessed 20 March 2018].

Valve. (2004) Hammer [Software]. Washington: Valve. Available from https://developer.valvesoftware.com/wiki/SDK_Installation [accessed 21 April 2018].

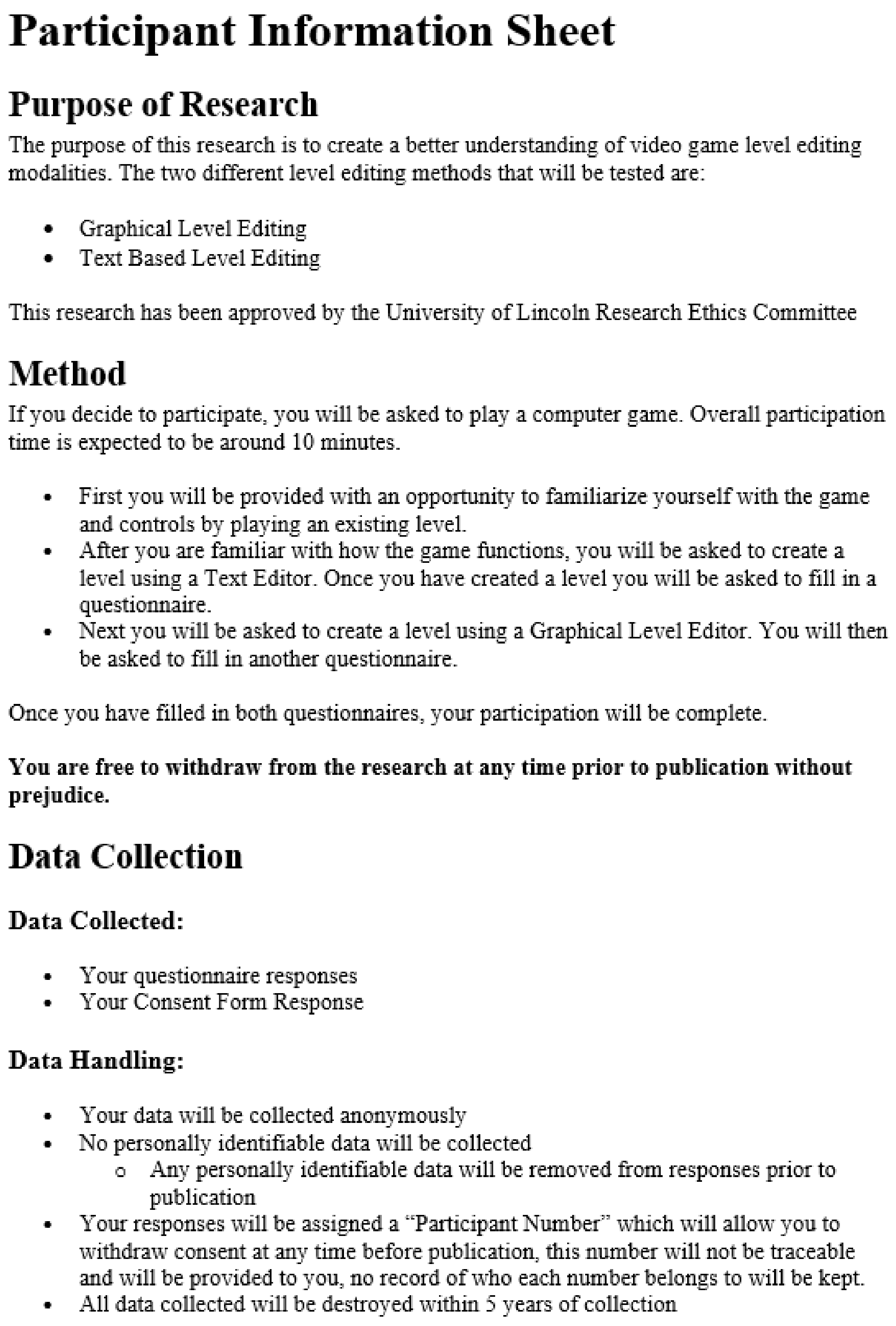
Xamarin. (2018) MonoDevelop [software]. Available from https://www.monodevelop.com/ [accessed 20 April 2018].

yWorks. (2017) yEd [software]. Available from https://www.yworks.com/products/yed [accessed 25 April 2018].

Figures

Appendices

a)



Participant Information Sheet

Purpose of Research

The purpose of this research is to create a better understanding of video game level editing modalities. The two different level editing methods that will be tested are:

- Graphical Level Editing
- Text Based Level Editing

This research has been approved by the University of Lincoln Research Ethics Committee

Method

If you decide to participate, you will be asked to play a computer game. Overall participation time is expected to be around 10 minutes.

- First you will be provided with an opportunity to familiarize yourself with the game and controls by playing an existing level.
- After you are familiar with how the game functions, you will be asked to create a level using a Text Editor. Once you have created a level you will be asked to fill in a questionnaire.
- Next you will be asked to create a level using a Graphical Level Editor. You will then be asked to fill in another questionnaire.

Once you have filled in both questionnaires, your participation will be complete.

You are free to withdraw from the research at any time prior to publication without prejudice.

Data Collection

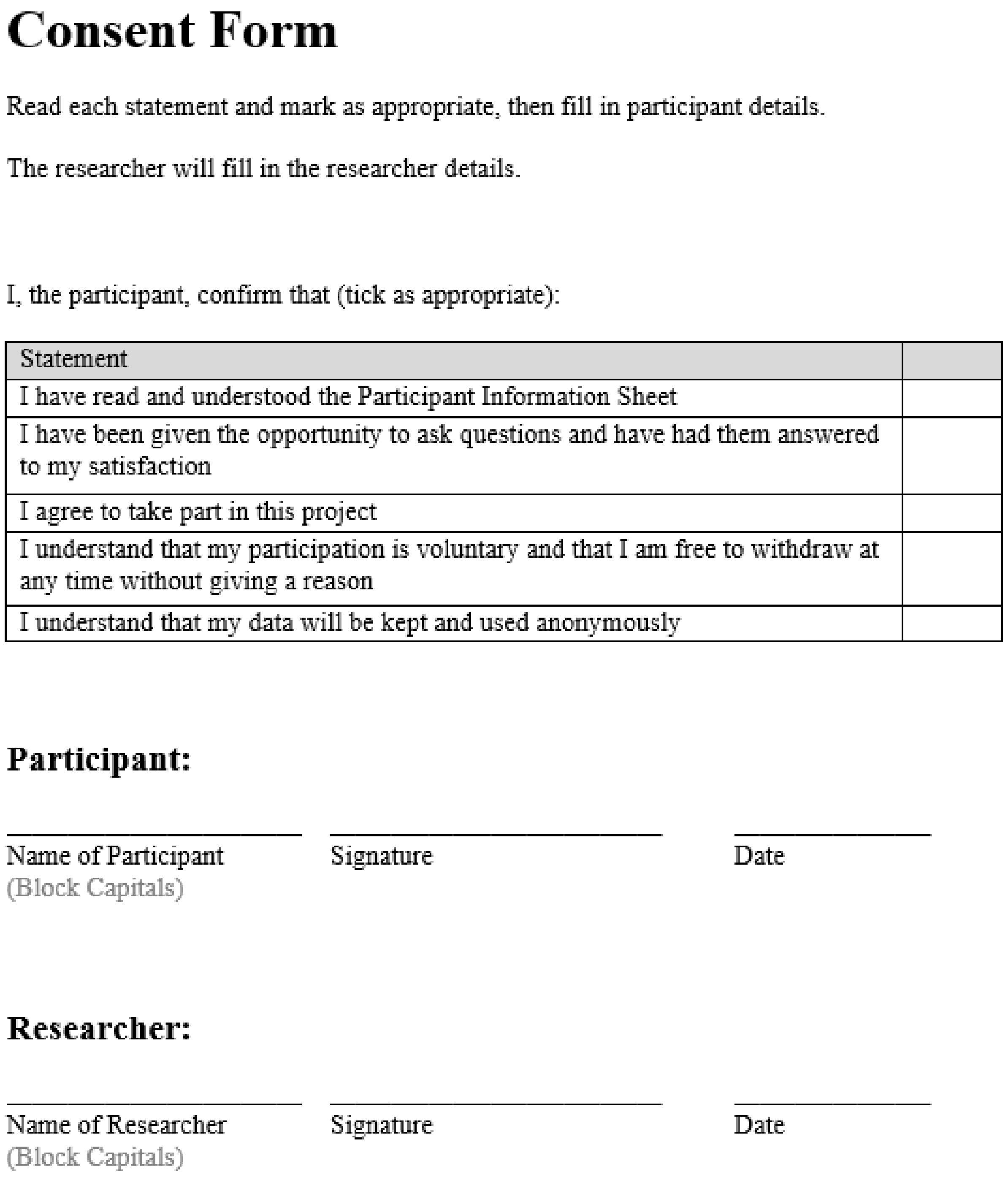
Data Collected:

- Your questionnaire responses
- Your Consent Form Response

Data Handling:

- Your data will be collected anonymously
- No personally identifiable data will be collected
    - Any personally identifiable data will be removed from responses prior to publication
- Your responses will be assigned a "Participant Number" which will allow you to withdraw consent at any time before publication, this number will not be traceable and will be provided to you, no record of who each number belongs to will be kept.
- All data collected will be destroyed within 5 years of collection

b)



Consent Form

Read each statement and mark as appropriate, then fill in participant details.

The researcher will fill in the researcher details.

I, the participant, confirm that (tick as appropriate):

Statement	
I have read and understood the Participant Information Sheet	
I have been given the opportunity to ask questions and have had them answered to my satisfaction	
I agree to take part in this project	
I understand that my participation is voluntary and that I am free to withdraw at any time without giving a reason	
I understand that my data will be kept and used anonymously	

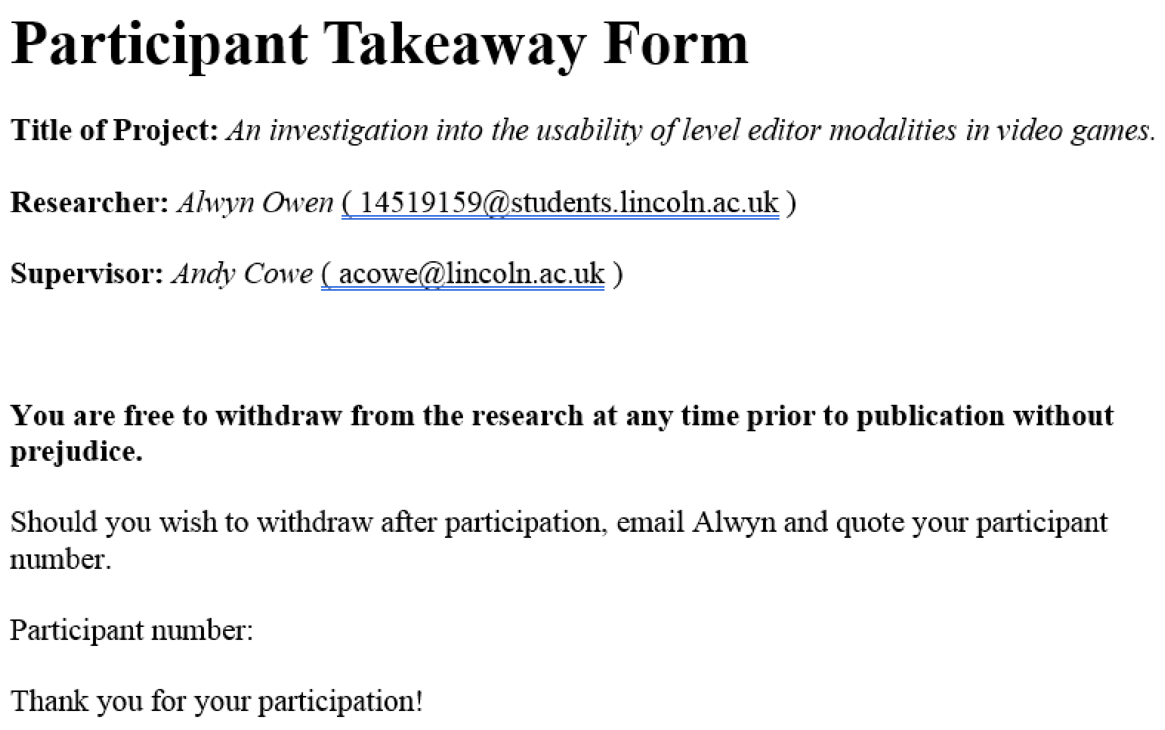
Participant:

____________________	______________________	____________
Name of Participant (Block Capitals)	Signature	Date

Researcher:

____________________	______________________	____________
Name of Researcher (Block Capitals)	Signature	Date

c)



Participant Takeaway Form

Title of Project: *An investigation into the usability of level editor modalities in video games.*

Researcher: *Alwyn Owen* (14519159@students.lincoln.ac.uk)

Supervisor: *Andy Cowe* (acowe@lincoln.ac.uk)

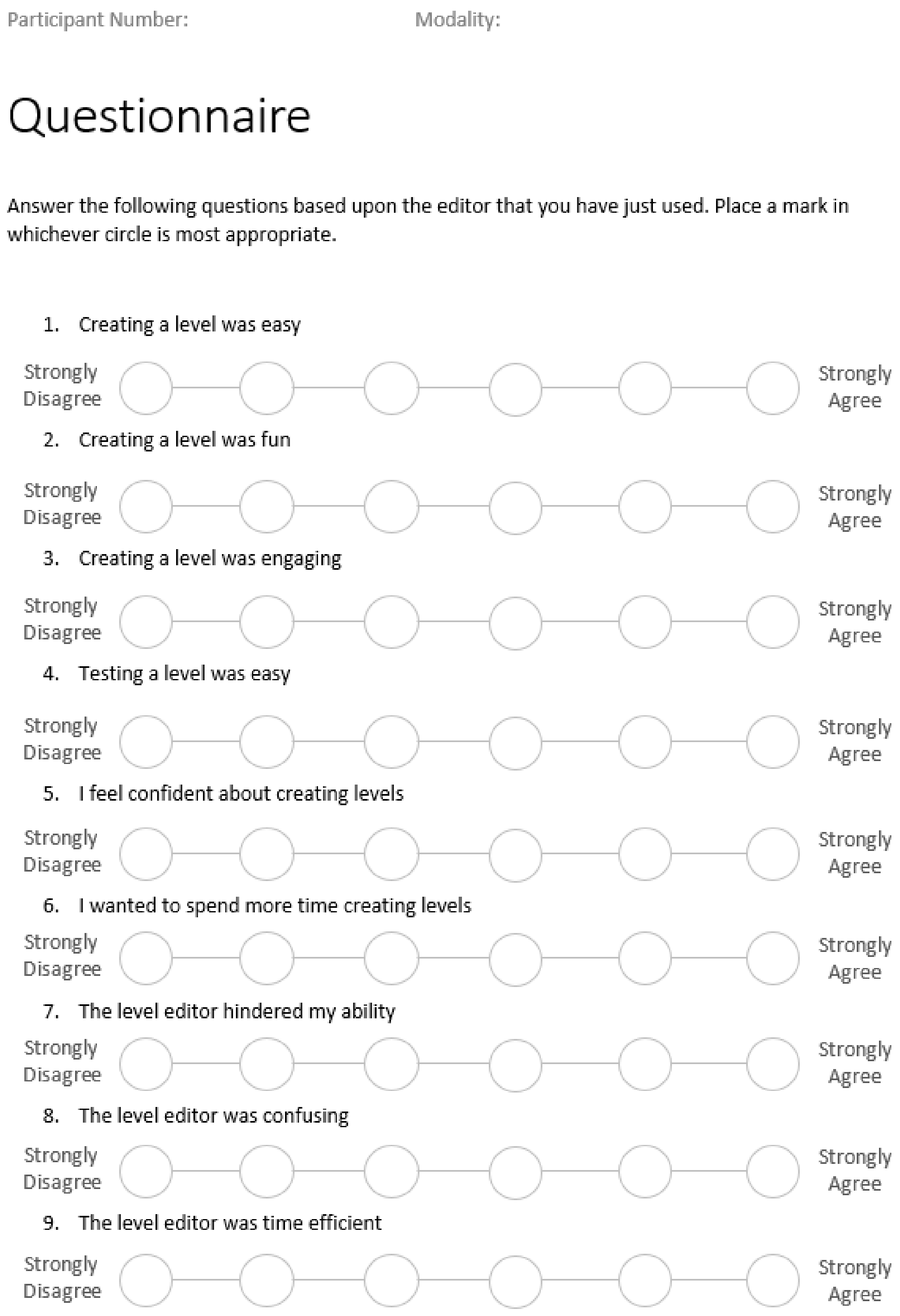
You are free to withdraw from the research at any time prior to publication without prejudice.

Should you wish to withdraw after participation, email Alwyn and quote your participant number.

Participant number:

Thank you for your participation!

d)



Participant Number: Modality:

Questionnaire

Answer the following questions based upon the editor that you have just used. Place a mark in whichever circle is most appropriate.

1. Creating a level was easy

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

2. Creating a level was fun

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

3. Creating a level was engaging

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

4. Testing a level was easy

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

5. I feel confident about creating levels

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

6. I wanted to spend more time creating levels

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

7. The level editor hindered my ability

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

8. The level editor was confusing

Strongly Disagree ○—○—○—○—○—○ Strongly Agree

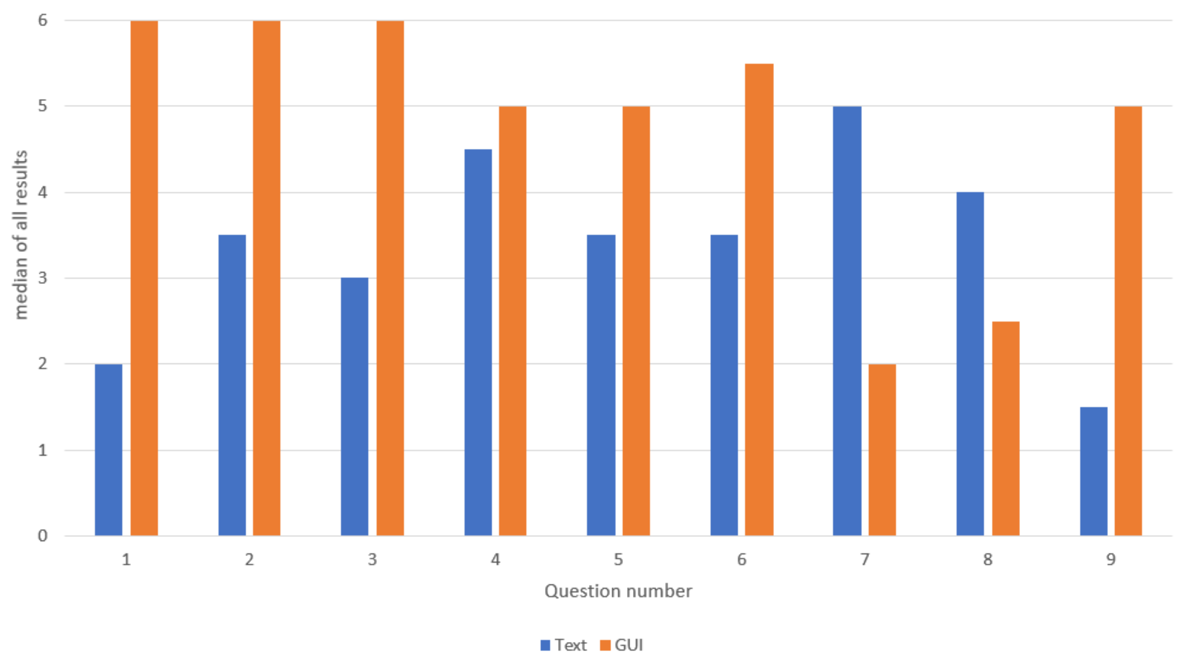
9. The level editor was time efficient

Strongly Disagree ○—○—○—○—○—○ Strongly Agree e)

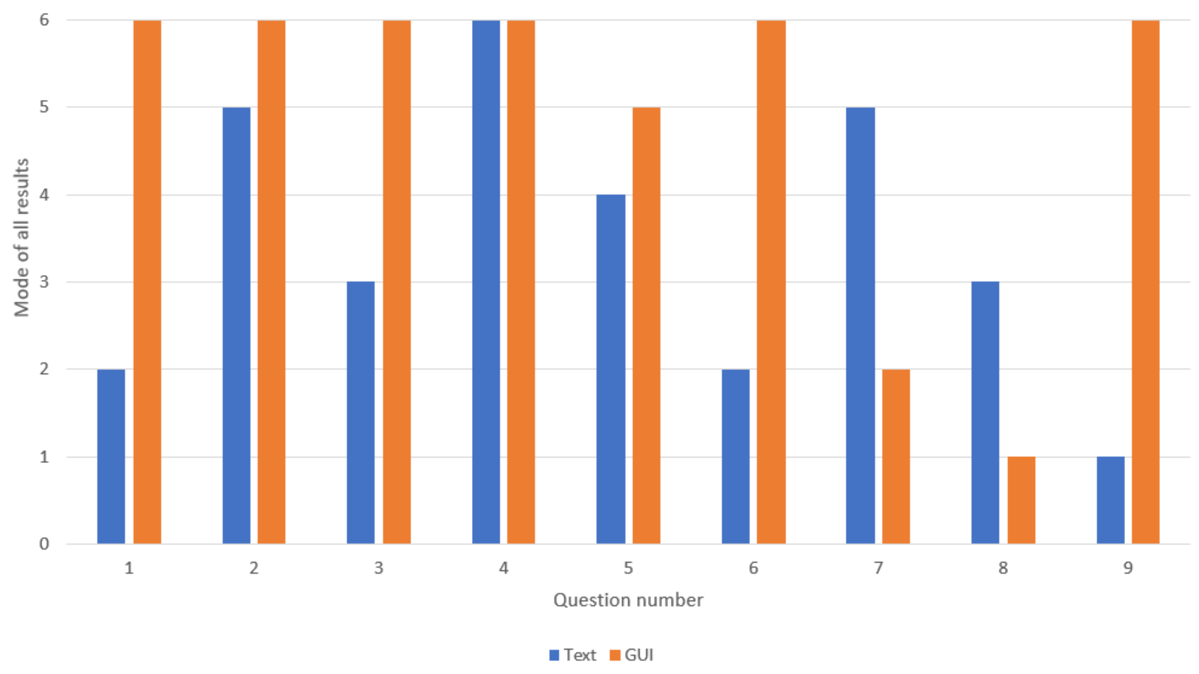
Text Editor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum	Mean	Median	Mode
Creating a level was easy	5	1	2	1	2	2	2	1	4	5	3	4	4	2	38	2.714285714	2	2
Creating a level was fun	5	5	2	6	2	1	3	1	4	5	3	5	5	2	49	3.5	3.5	5
Creating a level was engaging	3	3	2	6	1	2	3	1	5	6	3	6	5	2	48	3.428571429	3	3
Testing a level was easy	5	1	5	6	4	3	2	3	5	6	4	6	6	3	59	4.214285714	4.5	6
I feel confident about creating levels	4	2	1	4	2	3	3	1	4	4	4	5	4	3	44	3.142857143	3.5	4
I wanted to spend more time creating levels	4	2	1	6	2	3	4	1	5	6	3	5	5	2	49	3.5	3.5	2
The level editor hindered my ability	2	6	6	5	5	5	5	6	5	2	4	2	4	5	62	4.428571429	5	5
The level editor was confusing	3	6	5	3	5	4	4	6	5	4	5	2	3	3	58	4.142857143	4	3
The level editor was time efficient	1	1	1	1	1	2	2	1	3	6	3	6	4	1	33	2.357142857	1.5	1
GUI Editor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum	Mean	Median	Mode
Creating a level was easy	5	5	4	6	6	6	5	6	5	6	6	6	6	5	77	5.5	6	6
Creating a level was fun	5	4	5	6	6	6	3	6	6	6	6	6	6	5	76	5.428571429	6	6
Creating a level was engaging	4	5	4	6	6	6	4	6	6	6	5	6	6	5	75	5.357142857	6	6
Testing a level was easy	6	5	6	6	4	5	5	5	6	4	5	6	6	5	74	5.285714286	5	6
I feel confident about creating levels	5	5	5	6	5	6	5	4	5	6	6	6	5	6	75	5.357142857	5	5
I wanted to spend more time creating levels	6	5	4	6	5	5	3	4	6	6	6	6	6	5	73	5.214285714	5.5	6
The level editor hindered my ability	2	4	3	1	2	2	3	2	4	1	3	1	2	2	32	2.285714286	2	2
The level editor was confusing	1	3	3	2	1	5	3	3	5	1	5	1	1	2	36	2.571428571	2.5	1
The level editor was time efficient	4	5	5	6	6	6	5	6	4	4	5	6	6	5	73	5.214285714	5	6

Text Editor	22	3	3	22	4	7	10	-3	20	32	14	33	26	7
GUI Editor	32	27	27	39	35	33	24	32	29	36	31	40	38	32

f)



Graph showing Median responses to each questionnaire per question.



Graph showing Mode responses to each questionnaire per question.