Bachelor of Science in Digital Game Development
February 2018

Implementing a Level Design Tool for Calculating and Tuning the Travel Time of Paths in a Digital Game

Jesper Eriksson

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Digital Game Development. The thesis is equivalent to 10 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author:
Jesper Eriksson
E-mail: jeea14@student.bth.se

University advisor:
Dr. Yan Hu
Department of Creative Technologies

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. In the game Counter Strike: Global Offensive (CSGO), a noticeable amount of the levels in the competitive rotation is originally made by fans. The community of fan designers offers rule of thumbs regarding the travel time between key elements to ease the creation of a functional level. However, in the Hammer editor, where levels for CSGO are created, the level designers are using a general world building workflow consisting of the following steps: Load the level, test-run from point A to point B while watching the in-game round timer, exit the playtesting and reshape the geometry based on the timing feedback received from the test-run. In this thesis that workflow will be refered to as *Run & Time*.
Objectives. In this thesis, a new workflow, the *Timing Tool*, is introduced, specialized on following the rule of thumbs with the use of a custom tool. In the *Timing Tool* workflow, the designer types the desired travel time as an input to the tool. The tool then outputs a path of correct length and lets the designer connect its start and end point to the key elements. Thereafter, this study evaluates if the *Timing Tool* is more efficient and satisfying compared to a simulation of the already existing industry workflow, *Run & Time*, when it comes to calculate the travel time of connecting paths between key elements.
Methods. A user study with 10 participants was conducted in a controlled environment. Each particpiant performed a task using the two different workflows in turn. The task was to create a simplified Counter Strike: Global Offensive level, consisting of two key elements with connecting paths between them. The requirements of the level was that the connecting paths between the key elements had been tuned to match the travel time offered in the rule of thumbs given by the community. Immediately after finishing the task using one of the workflows, a satisfaction questionnaire was presented. The task time, a binary value of success or failure, and the answers from the questionnaire was used to evaluate efficiency and satisfaction by standards offered by the ISO/IEC and industry standards for user centered design.
Results. The *Timing Tool* workflow received a goal per minute score of 0,087 from the time based efficiency evaluation, more than twice as high as the *Run & Time* workflow which received 0,03. The overall efficiency evaluation gave the two workflows the same percentage since none of the participants failed a task. The system usability score gave the *Timing Tool* 82,78 (grade A) whereas *Run & Time* received a score of 71,94 (grade C).
Conclusions. The results showed that the *Timing Tool* was both more efficient and more satisfying than *Run & Time*. The results are however questionable since the study only had data from 9 participants which can bias the results. Also since *Run & Time* was not performed in the Hammer editor, the experiment could not perfectly replicate the workflow used during current CSGO map creation. However since there seems to be a sizable difference between the methods, future development of similar tools look promising.

Keywords: Balance, Guidelines, Usability Evaluation, CS:GO, Workflow.

Acknowledgments

I want to thank Michael Hüll, Dr. Yan Hu, the students who helped me piece this thesis together and the participants attending the test.

Contents

Chapter 1

Introduction

The word level can mean a lot of things in the video game industry, in this thesis it's defined as an environment or location where players interact with the game. In the game Counter-Strike Global Offensive, two teams, the Counter-Terrorists (CT) and the Terrorists (T), are competing against each other on a level also known as a map. To prevent one of the two teams from being more preferable than the other in terms of win percentage, the levels need to be balanced, meaning that the level has been tuned in the creation phase (Newheiser, 2009).
To quicken the process of creating a balanced map, the design community have come up with rule of thumbs and guidelines regarding the timing between key elements on the level (Ross, 2014). Meaning, the time it takes for the player character to run at maximum speed from one key element to another.

1.1 The Problem

Ernest Adams writes in his book about game design that, defining and tuning the gameplay is one of the most difficult tasks in designing a game (Rollings & Adams, 2003). The current way for the designer to follow the rule of thumbs, demonstrated by Alex Galuzin, consist of a series of time consuming and awkward steps that could be circumvented (*Run & Time*) (Galuzin, 2018). What could be the reason for not finding design tools or attempts of redefining the workflow is the separation between the Hammer editor and the Source 2 game engine.
CS:GO is a competitive game played in the tournaments of eSport. When looking at todays size of eSport in terms of spectators, competitors, yearly tournaments and the amount of money involved, the need for balanced levels gets highlighted (Li, 2017) (Ross, 2014). Since it's the community of fan designers that creates a noticeable amount of the competitive maps used in eSport, it seems wise that iteration time of the tuning shall be as efficient and satisfying for the designer as possible (Li, 2017). To ease the the level designers journey from beginner to quality CS:GO level creator.

1.2 Aim and Objectives

The aim of the thesis is to introduce a more efficient and satisfying workflow in terms of calculating and tuning the travel time of connecting paths between key elements

in the level creation phase of a competitive team-based multiplayer fps. To achieve this, a series of objectives had to be completed:

- Develop a custom design tool in Unreal Engine 4 (UE4)
- Develop a simulated version of the *Run & Time* workflow in UE4
- Gather test users that fulfill the participant criteria
- Design a user instruction session for two different workflows
- Design an experiment in UE4 to test two different workflows
- Evaluate the gathered data with the use of usability equations
- Compare the results on efficiency and satisfaction

1.3 Research Questions

RQ1: Which workflow is most efficient when it comes to calculate the travel time of connecting paths between key elements in a competitive team-based multiplayer fps?

RQ2: Which workflow is by the users perspective most satisfying to work with?

Chapter 2

Background

2.1 Counter-Strike: Global Offensive

Counter-Strike: Global Offensive referred to as CS:GO is the fourth game in the Counter-Strike series and was released 2012 by Valve Corporation. The first Counter-Strike game was released 1999 and became the most played online PC action game in the world almost immediately after its release. For the past 19 years, it has continued to be one of the most played games in the world, headline competitive gaming tournaments and selling over 75 million units worldwide across the franchise[1][2].
CS:GO is a competitive team-based action game, meaning that players compete in teams in attempt to fulfill a specific goal. The team that succeed in this are claimed winners (Rollings & Adams, 2003). It can also be described as an objective-based, multiplayer first-person shooter (FPS). FPS is a game genre centered around weapon-based combat (in this case guns) in a first person perspective. The first person perspective refers to a graphical perspective rendered from the viewpoint of the player or game character. The game is played in an online fashion, meaning that players are connected over the internet and can play with each other over far distances.

Cliffe, the co-creator of Counter-Strike, worked with the community of players, which submitted their own level designs that were incorporated into the game (Li, 2017).

eSports consists of different leagues, ladders and tournaments that's associated with competitive video gaming. Players like in sports, can choose to enter teams or other "sporting" organizations which are sponsored by various business organizations(Li, 2017)(McCutcheon, Hitchens, & Drachen, 2018). CS:GO is one of the games in eSport.

[1]No Author, VALVE ANNOUNCES COUNTER-STRIKE: GLOBAL OFFENSIVE (CS: GO), https://store.steampowered.com/news/6059/ (accessed 2018-05-17)

[2]Sergey Galyonkin, Counter-Strike: Global Offensive, http://steamspy.com/app/730 (accessed 2018-05-17)

2.2 Definitions of balance

There is no clear definition of balance, however there's a variety of interpretations. Since game designers view balance as an aesthetic process that integrate game elements, balance is seen as a quality of the game design and thus harder to quantify. Examples of game elements is level design, starting conditions and attributes to optimize for stability and enjoyment (Oxland, 2004)(Rollings & Morris, 2004)(Novak, 2012). Mark Newheiser interprets balance as the concept and the practice of tuning a game, usually with the goal of preventing a component system to be more preferable when compared to their peers (Newheiser, 2009). Ernest Adams interprets balance as a tuning process aimed at removing dominant strategies to give each player a chance of success. The process differs between games and while there are no standards, there are general guidelines to support it (Adams, 2014). The rule of thumbs described in the section below is one example of such guidelines offered by the CS:GO community. At its simplest, a balanced game is considered to be a game with no dominant strategies available (Oxland, 2004)(Rollings & Morris, 2004).
Owen Makin and Shaun Bangay decided to use the similar definition "a balanced game is one with no dominant strategies" in their science paper. They consider balance to be property of the game itself and thus independent of player skill (Makin & Bangay, 2017). For this paper, the same definition is used, with the inclusion that a certain "team pick" could be considered a dominant strategy if it's a higher percentage of winning by being in that team on the particular level that is about to be played.

2.3 Rule of Thumbs

In CS:GO, two teams, the Counter-Terrorists (CT) and the Terrorists (T), are competing against each other on a level also known as a map. To prevent one of the two teams from being more preferable than the other in terms of win percentage, the levels need to be balanced, meaning that the level has been tuned in the creation phase (Newheiser, 2009). To quicken the process of creating a balanced map, the design community have come up with rule of thumbs and guidelines regarding the timing between key areas on the level (Ross, 2014). Meaning, the time it takes for the player character to run at maximum speed from one key area to the other. The two key elements in the simplified level are:

- Spawn areas:
  The location of a teams starting position.

- Bombsites:
  The areas where the bomb can be planted (in the CSGO demolition game mode).

And the rule of thumbs regarding the timing between the above mentioned key elements are:

- The time between T spawn and the bombsites: 30 seconds.

- The time between CT spawn and the bombsites: 15 seconds.
  - The defending team must reach the bombsites in a faster time than the attacking team, so that they can get into defending position before the battle starts.
- The time between CT spawn area and T spawn area: 45 seconds.
- The time between the two bombsites: 20 seconds.
  - The bomb timer is 30 seconds and the disarming time is 10 seconds. Therefore it should take no longer than 20 seconds for the player to reach the bomb when they hear it's been set.

The motivation behind the specific time values is not clear. Galuzin uses similar times and states that he got them from measuring the official maps (Galuzin, 2018). However, the creator of Mirage, which is one the official maps, was not fond of these specific times. But after measuring his own maps and the official map dust2, similarities were found. The time between the key elements differed with a maximum of two seconds. He pointed out that when following the rule of thumbs, there's no guarantee that the map will be any good or balanced, but it could be a good exercise for beginners to get an understanding of distances in the editor and the general design of a CS:GO map. More regarding the subject is discussed in the discussion chapter.

2.4 Valve Hammer Editor

Valve Hammer Editor, formerly known as Worldcraft and commonly named Hammer, is Valve Software's map creation program for their game engine, Source. It is the program used to create CS:GO levels and it is freely available to anyone who has purchased a Source based game e.g. Counter-Strike, Half-Life.
It was first released in 1996 by Ben Morris, valve acquired the engine after hiring Ben for the development of Half-Life in 1997. Counter-Strike was initially created as a modification, or mod, by using existing assets from Half-Life. It was created by two college students, Minh "Gooseman" Le and Jess Cliffe. Valve hired Le and Cliffe and gave Counter-Strike a commercial release in 1999 (Li, 2017).

2.5 Unreal Engine 4

The Unreal Engine is a game engine developed by Epic Games, first showcased in the game fps game Unreal 1998. The tactical shooter Tom Clanscy's Rainbow Six Siege from 2015 is one example of a modern game using Unreal Engine. Although primarily developed for fps, it has been successfully used in a variety of other genres (Shekar & Karim, 2017).
There are three main reasons for picking UE4 to create the tool for this study:

- Included Level Editor:
  Because the game engine has an included level editor, the process of creating the tool became easier, since the delay from thought to action became shorter.

It was now possible to both get quick feedback of the code and iterate on new ideas in a faster pace because of the visual feedback received from the level editor.

- Large Community of Developers:
  In 2014, at the Game Developers Conference, Epic Games released Unreal Engine 4 to the development community through a new subscription model. Since the engine became free to use, the amount of developers increased. Developers and content creators can sell their assets and tools using the Unreal Engines marketplace[3].

- Wider Usage Area:
  It would ease the process to in the future, test the tool in different types of games and genres.

2.6 Alex Galuzin

Alex Galuzin is a level designer respected in the CS:GO community who has been writing articles and creating tutorials on his website for FPS games using both Unreal Engine and the Valve Hammer Editor for 9 years. Galuzin demonstrates his workflow in one of his videos regarding the creation process of a bomb defusal level in the game CS:GO (Galuzin, 2018). It can be broken down into these steps:

- Draw a top down layout sketch of the level
- Block out the playable space of the level with simple geometrical shapes
- Calculate and tune the travel time of the connecting paths between the above mentioned key elements in accordance with the rule of thumbs.

2.7 The Run & Time Workflow

This thesis will be focusing on the last step of Galuzin's workflow, to calculate and tune the travel time of connecting paths between a levels key elements in accordance with the rule of thumbs (Galuzin, 2018). This is done in the following steps:

- Load the level by clicking play
- Test-run from point A to point B while watching the in-game round timer
- Exit play
- Reshape the simple geometry based on the timing feedback received from the test-run.

[3]Unreal Engine, Marketplace, https://www.unrealengine.com/en-US/faq (accessed 2018-05-17)

2.8 The Timing Tool Workflow

The new workflow, implemented in this thesis, will be named *Timing Tool*. The steps of the *Timing Tool* is demonstrated below:

- Inside the game engine, the designer drags out the tool into the playable space of the level.
- As the designer selects the tool, a text field appears. The designer can then type in the desired time between point A to point B to the tool as an input. The tool responds by outputting a surface with a fixed distance in the shape of a dotted line. If a player were to travel from start to end of that dotted line by running, it would result in the desired time given as an input.
- The designer places the dotted line in between where the measurement is to be made, and modifies the path to fit without interfering with the travel time.

2.9 The Shelling System

The Shelling System, made by Alessa Baker is an UE4 design tool, used to block out the basic look of a level. It was similar to the workflow used in the Hammer editor and were therefore used in this thesis to simulate the *Run & Time* workflow in UE4. The Hammer editor workflow has three key functionality steps which all could be simulated in UE4 using the Shelling System, the steps were the following:

- **Duplicate:** In the Hammer editor, the designer often duplicate geometry if a similar shape is to be made. that step is supported by the Shelling System.
- **Move:** In both the Hammer editor and UE4, the designer can choose to move and entire geometrical objects, that action is done in the same way, using the same hotkeys.
- **Scale:** Objects are scaled in a similar way in the Hammer editor and the Shalling system. The designer selects an objects control point and can there after change the size by dragging.

However, there are qualities that differed between the real workflow and the simulated one, those are:

- **Orthographic View:** The designer can use the ortographic views in the Hammer editor to ease the way to select and shape objects. UE4 has orthographic views but they were not presented for the participants during the experiment.
- **Camera Controls:** The Hammer editor lets the designer zoom with the scroll wheel and move around in the scene by right click and drag. In UE4, the designer moves around with the W,A,S and D keys, commonly used in games using the first person camera perspective. That system lets the designer get close to objects by moving closer to it, instead of scrolling as in the Hammer editor.

- **Control Points:** Objects in the Hammer editor has a control point in every corner and every mid-point of it's bounding box. The Shelling System has instead one smart control point in one of the corners, it's smart because it can be used for both scale and move, depending on the mouse-dragging direction.

Chapter 3

Method

The thesis research method, follows the standards offered by ISO / IEC and Industry Standards for User Centered Design (Mifsud, 2015)[1]. The method consists of an implementation, an experiment and a questionnaire. The implementation is made to introduce the new workflow. The experiment, to get data for the efficiency evaluation and something the users can base their satisfaction upon. The participant criteria for partaking in the experiment is to have at least 20 hours gameplay experience of the game CS:GO; to lower the challenge of the experiment and to simulate a CS:GO level design beginner. The questionnaire is used to get data for the satisfaction evaluation.

3.1 The Implementation

A tool for calculating the running time of a path was implemented, to introduce the *Timing Tool* workflow. It was created in UE4 to meet a large community and to be future tested in a wider area with different types of games. The tool was built using a visual scripting system inside of UE4 called blueprints to circumvent as much math and implementation time as possible. It built on top of the already existing Spline Components that Unreal Engine offers through the blueprints. A spline function is a curve constructed from polynomial segments that are subject to conditions or continuity at their joints. Four additional features was added for this thesis.

- Offer control points for the designer to use, granting the ability to bend the spline.
  - **The Transforms:** Used for two reasons. The first, to disconnect the spline from it's endless length function. The second, to grant the ability for the designer to shape the look of the spline mid way.

- The functionality for a spline to be of a fixed length.
  - **The Calculate Length Function:** Used to measure the travel time from start to end of the spline. It's done by using each control point's location in object space to create a vector in between each point. With the information about the player characters speed, the length is divided by the movement speed, resulting in travel time. If the calculated travel

[1]ISO/IEC, TR 9126-4:2004, https://www.iso.org/standard/39752.html (accessed 2018-06-05)

time is less than the desired time given as an input, the push function is called, or if it's more, the pull function is called.

  - **The Push Function:** Used to move a control point in a direction away from its neighbor point containing a lesser index.

  - **The Pull Function:** Pulls a control point away from its neighbor point containing a higher index.

- The path mesh surface.

  - **The Procedural Mesh Function:** The designed can enable this to test if the vertical curvature is too steep for the player character to run upon.

- The time, displayed above the control points

  - **The Text Render Component:** granting the visual feedback of time, mid-way of the spline. So that placement of objects and elements does not necessary have to be placed and the start or the end points of the path.

3.2 Experiment Procedure

The experiment was taking place in a controlled environment where only the researcher and the participant were present. Before starting the task, each participant had a five minutes play session where they got to know the two workflows and UE4. Thereafter, each participant performed a task using each workflow in a controlled environment. The task was to create a simplified CSGO level, consisting of two key elements, found in section 2.3 with connecting paths between. The levels requirement was that the connecting paths between the key elements had been tuned in order to match the travel time offered in the rule of thumbs found in section 2.3.
The experiment was using a counterbalanced starting order which let 5 participants begin the experiment using the *Run & Time* workflow and 5 participants begin the experiment using the *Timing Tool* workflow which removes potential confound. Immediately after finishing the experiment using one of the workflows, a satisfaction questionnaire were presented. The task time for each workflow was measured by the screen recording software called Open Broadcaster Software (OBS).

3.3 The Participants

All the participant that attended the experiment were approximately around 20 to 28 years of age and were familiar with computer games. However, no data of such were gathered.

3.3.1 Participant Criteria

The participant criteria for partaking in the experiment is to have at least 20 hours gameplay experience of the game CS:GO; to lower the challenge of the experiment and to simulate a CS:GO level design beginner. The participants had a clear explanation about the terminology used in the experiment.

3.3.2 The Invitation Letter

The invitation letter was sent to the students of BTH but also to individuals the researcher assumed fulfilled the participant criteria. The letter invited the individuals to partake the usability study if they fulfilled the participant criteria which was described directly underneath. Thereafter came the purpose of the study and a small description of what the individual would do when partaking the experiment and how long time the experiment would take. Lastly, the location to the experiment room was presented as well as a link to a site where the individual could book their time.

3.4 The Efficiency Evaluation

The ISO 9241-11 standard defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. The definition states that usability is not a single, one-dimensional property but rather a combination of factors (Mifsud, 2015). This thesis leaves effectiveness to be measured in a future work because of the tight time schedule. Efficiency, according to the ISO/IEC 9126-4 Metrics is defined as the following "The resources expended in relation to the accuracy and completeness with which users achieve goals"[2].

3.4.1 Time Based Efficiency

Time Based Efficiency is measured in terms of task time and a binary value where 1 equals success and 0 equals failure. Task time is the time participant takes to successfully complete a task, measured in seconds or minutes. The binary value 1 is assigned if the test participant manages to complete the task and 0 if the participant fails to complete the task. The task is considered considered completed if the travel time of the connecting path is differing less than 1 second from the corresponding rule of thumb time. The formula for calculating time based efficiency will be the following:

$$Time\ Based\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR}$$

[2]ISO/IEC, TR 9126-4:2004, https://www.iso.org/standard/39752.html (accessed 2018-06-05)

Where:
N = The total number of tasks.
R = The number of users
Nij = The result of task i, by user j.
Tij = The time spent by user j to complete task i.

3.4.2 Overall Relative Efficiency

Overall Relative Efficiency also requires task time and the binary value earlier described in section 3.4.1. This calculation is using the ratio of the task time taken by the users who successfully completed the test in relation to the total time taken by all users. The formula for calculating overall relative efficiency, using the same variable explanation as the one above, is the following:

$$Overall\ Relative\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{N} t_{ij}} \times 100\%$$

3.5 The Satisfaction Evaluation

Satisfaction takes part as a factor of what defines usability in the ISO 9241-11, earlier mentioned in the efficiency evaluation section 3.4. ,Satisfaction is defined by the ISO/IEC 9126-4 Metrics as "The comfort and acceptability of use"[3]. User satisfaction is measured through standardized satisfaction questionnaires which can be administered after each task and/or after the usability test session (Mifsud, 2015). The standardized satisfaction questionnaire used in this study is the System Usability Scale Questionnaire.

3.5.1 The System Usability Scale Questionnaire

In 1986, John Brooke published the concept of a “System Usability Scale (SUS),” a “reliable, low-cost usability scale that can be used for global assessments of systems usability.” SUS is based on a Likert scale questionnaire with standardized content that gives an overall usability and user satisfaction index (ranging from 0 to 100). SUS is technology independent and has since been tested on hardware, consumer software, websites, cell-phones, IVRs and even the yellow-pages. It has become an industry standard with references in over 600 publications[4] [5] .

[3]ISO/IEC, TR 9126-4:2004, https://www.iso.org/standard/39752.html (accessed 2018-06-05)
[4]meiert, Revitalizing SUS, the System Usability Scale, https://meiert.com/en/blog/revitalizing-sus-the-system-usability-scale/ (accessed 2018-06-05)
[5]measuringu, Measuring usability with the system usability scale (sus), https://measuringu.com/sus/ (accessed 2018-06-05)

3.5.2 Motivation for using The System Usability Scale

Compared to other tests, the System Usability Scale (SUS) doesn't require a lot of resources to administer. Because of the limited time budget, this type of survey was found useful to get good information fast. Instead of researching and designing a brand new questionnaire, the template was used.

3.5.3 Administration of the System Usability Scale

Participants ranks each question from 1 to 5 based on how much they agree with the statement they are reading. 5 means they strongly agree, 1 means they strongly disagree.

3.5.4 The SUS Score Calculation

The SUS score is calculated in the following way. For each of the odd numbered questions, subtract 1 from the score. For each of the even numbered questions, subtract their value from 5. These new values adds up to a total score. Then multiply this by 2.5. Resulting in a score between 0 to 100, which is not a percentage, but a clear way of seeing the score. The calculation is showed below in 3.1.

$$((Q1-1)+(5-Q2)+(Q3-1)+(5-Q4)+(Q5-1)+(5-Q6)+(Q7-1)+(5-Q8)+(Q9-1)+(5-Q10))*2,5 \tag{3.1}$$

3.5.5 Industry Standards for SUS Scores

The average SUS score from 500 studies is a 68. A SUS score above a 68 would be considered above average and anything below 68 is below average (Mifsud, 2015).

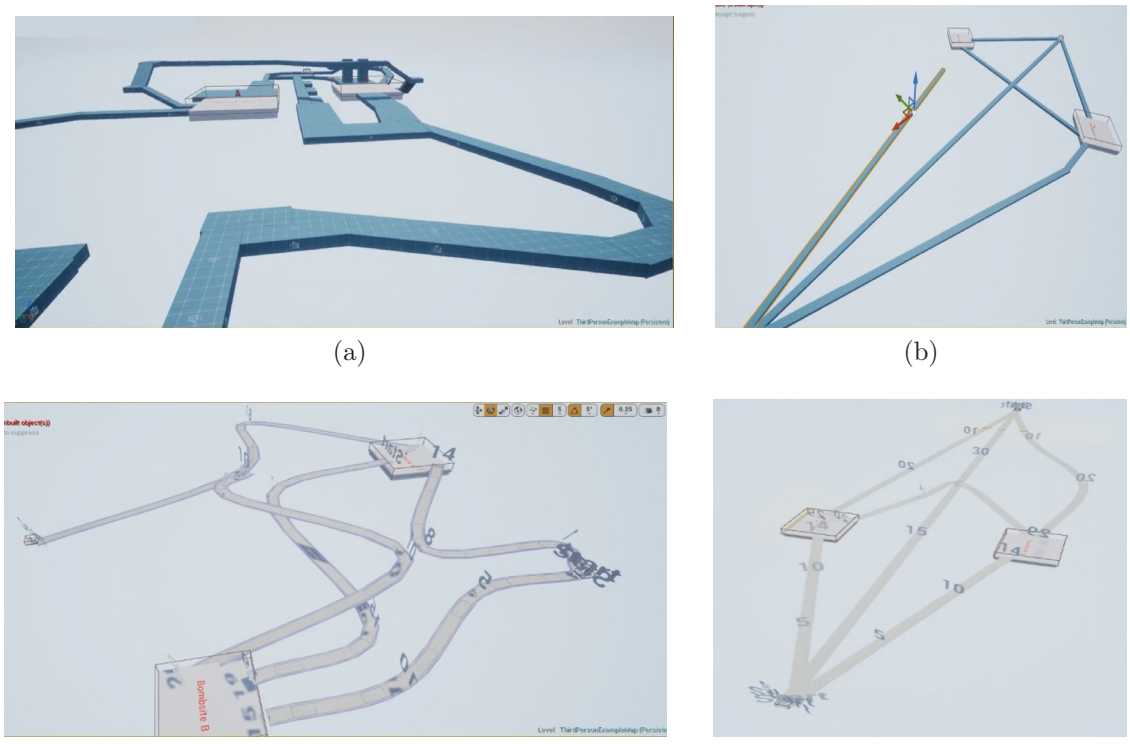
Here's an overview of how the scores should measure:

- 80.3 or higher is an A. The top 10% from the above-mentioned studies received this score. This is the point where users are more likely to recommend the system to their friends
- 68 or thereabouts is a C. 50% from the above-mentioned studies received this score.
- 51 or under, is an F. 15% from the above-mentioned studies received this score. It's recommended to make usability a priority and fix the system.

Chapter 4

Results

4.1 Examples of Successfully Completed Levels



(a) (b)

(c) (d)

Figure 4.1: Screenshots of two levels created during the experiment. a) A complex solution, using the *Run & Time* workflow. b) A simplistic solution, using the *Run & Time* workflow. c) A complex Solution, using the *Timing Tool* workflow. d) A simplistic solution, using the *Timing Tool* workflow.

The completed levels differed between each participant in complexity resulting in various task times. The figure 4.1 shows typical complex solution matched against more simplistic solution using the same workflow.

4.2 Task Time

Run & Time (Minutes)	Timing Tool (Minutes)
28,51	6,03
33,03	40,15
29,04	10,31
40,48	10,25
21,14	7,47
35,18	9,59
49	18,59
46,29	16,14
31,57	22,18

Figure 4.2: The participants task time is presented in the table

Task time, seen in table 4.2 is the time participant takes to successfully complete a task. It's interesting to note how the time differed between the two workflows but also how the time differed between the participant using the same workflow. The task time of participant 9, using the *Run & Time* workflow, is 46 minutes and 29 seconds, whereas participant 6, completed the same task using the same workflow in 21 minutes and 14 seconds. It's a 2.19 times difference, more about that in the discussion chapter at the subsection regarding diverging data 5.10.

4.3 Evaluation of efficiency

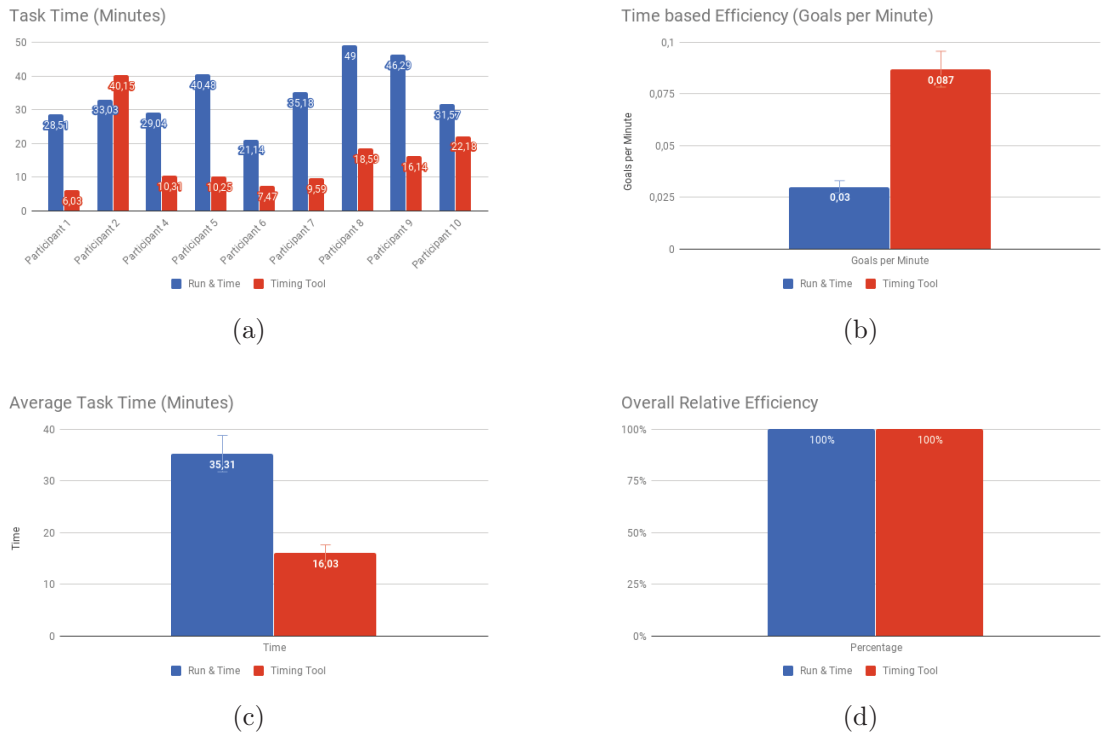


Figure 4.3: Efficiency results presented in stapel diagrams. a) The time it took for each participant to successfully finish the task, earlier called task time. b) The results from the timebased efficiency evaluation of the two workflows. c) Average time upon completion. d) Presentation of the overall efficiency results.

Participant 3 was discarded from the test due to an error of execution. The participant calculated the time between the key elements without creating any paths. Due to that there is data from 9 participants instead of 10. In 8 out of 9 cases, participants finished the test in a faster time by using the new implemented the *Timing Tool* workflow, 4.3a. The time based efficiency equation gave the *Timing Tool* a higher goal per minute score than the *Run & Time* workflow. This means that the average participant, when using the *Timing Tool*, can complete calculating two levels in a faster time than what it would take to calculate one, using *Run & Time*, see 4.3b. The overalll relative efficiency equation gave *Run & Time* and the *Timing Tool* the same percentage because all the participants completed the test 4.3d. The *Timing Tool* have a faster average time, less than half the time of the *Run & Time* workflow, see 4.3c.

4.4 The Satisfaction Evaluation

The Satisfaction Questionnaire:

Participants	PQ	Workflows	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS Score
Participant 1	2	Run & Time	3	2	4	1	2	2	5	5	3	1	65
		Timing Tool	5	1	5	1	2	2	5	1	5	1	90
Participant 2	3	Run & Time	4	1	5	1	4	1	5	1	5	2	92,5
		Timing Tool	3	2	4	1	4	1	4	2	3	2	75
Participant 4	2	Run & Time	1	1	5	1	3	1	5	5	5	1	75
		Timing Tool	5	1	4	2	4	3	5	2	5	2	82,5
Participant 5	1	Run & Time	1	5	5	1	3	1	5	3	5	3	65
		Timing Tool	3	1	5	1	5	1	5	1	5	1	95
Participant 6	4	Run & Time	1	3	4	1	4	1	5	3	4	2	70
		Timing Tool	5	2	5	1	4	2	5	2	4	2	85
Participant 7	4	Run & Time	2	3	4	1	3	2	2	2	4	2	62,5
		Timing Tool	4	2	4	2	4	2	4	2	3	2	72,5
Participant 8	4	Run & Time	2	3	4	1	3	2	2	2	4	2	62,5
		Timing Tool	4	2	4	2	4	2	4	2	3	2	72,5
Participant 9	2	Run & Time	3	1	3	1	4	2	3	2	5	1	77,5
		Timing Tool	4	2	5	1	4	2	5	2	5	1	87,5
Participant 10	5	Run & Time	3	1	3	1	4	2	3	2	5	1	77,5
		Timing Tool	4	2	5	1	4	2	5	2	5	1	87,5

Table 4.1: The table shows each of the participants satisfaction questionnaire answers as well as their individual SUS score

The questionnaire is written in such way that the participant must think before answering a question. Strongly agree may on one question be a positive answer whereas on another, a negative. See the questions of the questionnaire in the appendix A.4. The equation for calculating a SUS Score is presented in the background chapter 3.1, it was used on each individual participant.

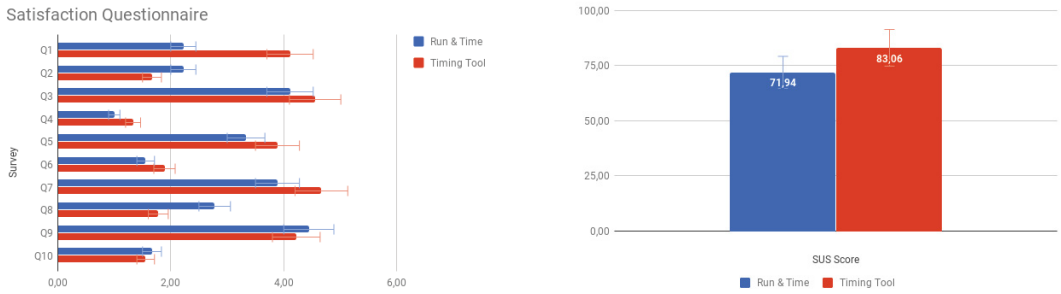
The Satisfaction Questionnaire Average Score:

Average	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS Score
Run & Time	2,22	2,22	4,11	1,00	3,33	1,56	3,89	2,78	4,44	1,67	71,94
Timing Tool	4,11	1,67	4,56	1,33	3,89	1,89	4,67	1,78	4,22	1,56	83,06

Table 4.2: The table shows each of the participants satisfaction questionnaire answers as well as their individual SUS score

4.4.1 Staple Diagrams:

The System Usability Scale: The new implemented *Timing Tool* got a 82,78 SUS score, which means it got the highest grade compared to industry standards. The *Run & Time* workflow scored 71,94 which is above average, meaning it's ok but could be improved. The SUS scores are presented in a diagram on 4.4b. The *Timing Tool* had an average more positive score on each and every question on the questionnaire, see 4.4a.



(a) (b)

Figure 4.4: Diagrams regarding the satisfaction questionnaire a) A score for each individual question of the questionnaire, b) The SUS score from the satisfaction questionnaire

Chapter 5

Analysis and Discussion

5.1 Average Time

The average time for each workflow can be calculated by taking the time from table 4.2. The results is the following:

- The *Run & Time* workflow: 35 minutes and 31 seconds.
- The *Timing Tool* workflow: 16 minutes and 3 seconds.

5.2 Timebased Efficiency Evaluation

By using the time in table 4.2, and the formula earlier presented in the method section, the time based efficiency can be calculated for each workflow:

- *Run & Time* workflow::

$$\frac{(\frac{1}{28.51} + \frac{1}{33.03} + \frac{1}{29.04} + \frac{1}{40.48} + \frac{1}{21.14} + \frac{1}{35.18} + \frac{1}{49} + \frac{1}{46.29} + \frac{1}{31.57})}{1 * 9} = 0.030 \; Goals \; per \; Minute \quad (5.1)$$

- The *Timing Tool* workflow:

$$\frac{(\frac{1}{6.03} + \frac{1}{40.15} + \frac{1}{10.31} + \frac{1}{10.25} + \frac{1}{7.47} + \frac{1}{9.59} + \frac{1}{18.59} + \frac{1}{16.14} + \frac{1}{22.18})}{1 * 9} = 0.087 \; Goals \; per \; Minute. \quad (5.2)$$

By looking at both the average time and the goal per minute result, one can see that the average participant, when using the *Timing Tool* workflow, can complete calculating two levels in a faster time than what it would take to calculate one, using *Run & Time*. A reason for why *Timing Tool* got better results in efficiency might be because it's specialized in the particular task the participants partook in the experiment. Whereas *Run & Time* is part of a general world building workflow.

5.3 Overall Relative Efficiency

By using the captured time at table 4.2 in the equation below 5.4, we get an overall relative efficiency percentage. Since all the participants completed the ask with both workflows, the equation 5.4 outputs a results of 100% Overall Relative Efficiency.

- The *Run & Time* workflow:

$$(\frac{(1*28.5)+(1*33)+(1*29)+(1*40.5)+(1*21.14)+(1*35.2)+(1*49)+(1*46.3)+(1*31.6)}{(28.5+33+29+40.5+21.14+35.2+49+46.3+31.6)})*100 = 100\%. \tag{5.3}$$

- The *Timing Tool* workflow:

$$(\frac{(1*6)+(1*40.2)+(1*30.3)+(1*10.3)+(1*7.5)+(1*9.6)+(1*18.6)+(1*16.1)+(1*22.2)}{(6+40.2+30.3+10.2+7.5+9.6+18.6+16.1+22.2)})*100 = 100\%. \tag{5.4}$$

A possible reason for why both workflows received an overall efficiency score of 100% might be because of the low amount of participants. For the percentage to differ between the two workflows, failure is needed, something that might have occurred if more participants partook the experiment.

5.4 The Satisfaction Questionnaire

When looking at the average questionnaire answers in the result section, the answers on Q1 and Q8 differed the most between the workflows.

- Q1 " I think that I would like to use this System frequently".
- Q8 " I found the system very cumbersome to use".

The *Run & Time* workflow received an average of 2,22. The reason might be because of the lengthy action the participants had to go through in order to get feedback to tune the paths. In order to know if the path between CT spawn and T spawn is 45 seconds, the participant had to run for at least 40 seconds in order to get an understanding of whether the path was too long or too short. It's arguable that pressing W (the key for moving forward) for 40 seconds, with nothing else to do, is a boring and cumbersome process. It's therefore understandable that the participant would not want to go through the process frequently. Whereas when using the *Timing Tool* workflow, which got an average of 4,11, there is constant action from the designers part. This might also contribute to why the *Timing Tool* received such high efficiency score.

5.5 The Average SUS Score

The individual SUS scores from table 4.2 were put in the equation below 5.5, resulting in a score of 71,94.

- The *Run & Time* workflow:

$$\frac{(65+92,5+75+65+70+62,5+62,5+77,5+77,5)}{9} = 71,94. \tag{5.5}$$

- The *Timing Tool* workflow:

$$\frac{(90+75+82,5+95+85+72,5+72,5+87,5+87,5)}{9} = 83,06. \tag{5.6}$$

Both of what was written in the time based efficiency section 5.2 and the discussion above regarding the answers given to question Q1 and Q2 5.4 are believed to have the most impact over why the *Timing Tool* was defined as more satisfying than the *Run & Time* workflow in this study.

2

5.6 The Maker of an Official Map, Michael "BubkeZ" Hüll,

After being in contact over email with Michael "BubkeZ" Hüll, the creator of the CS:GO level Mirage (aka de_cpl_strike), one of the arguably most popular and balanced maps in the game, new insights regarding the creation process has been made. Michael H was not following the workflow Alex Galuzin demonstrated on his website but could understand it to be a good exercise for beginners to getting a hang of distances in the hammer editor and the general design of a CS:GO map. But in his case, the experience of both playing the game on a high level and designing maps for a long time has given him the ability to sense the distances between the key elements directly in the creation phase. Giving the conclusion that both of the proposed workflows are superfluous once you've gotten to a certain level of expertise. After calculating the time between the key elements on the levels Dust2, Mirage and Michael's new map Iris, similarities were found regarding the time of the connecting paths, but not close to the times given by the design community. Michael's new set of rule of thumb times would be as following:

- The time between CT spawn area and T spawn area: 20 seconds.
- The time between CT spawn area and the bombsites: 10 seconds.
- The time between T spawn and the bombsites: 15 seconds.
- The time between the two bombsites: 15 seconds.

Michael wanted to point out that even if one were to follow rules of timing, there's no guarantee that the map will be any good or balanced, a strong understanding of high level gameplay and a pro-players requirements is needed for that.

5.7 The Discarded Participant

The experiment description did not state the definition of path. This participant was fast upon completion in both workflows, but his levels was missing visible paths between the key elements, which led him to be discarded. The travel time between the key elements was correct, but the level was made on an open flat plane.

5.8 The Experiment Time

In the invitation letter, it was stated that the test would take 35 minutes but was found to take on an average 58 minutes. This mistake could influence the quality of the data, considering the fact that the participants were not ready for such duration.

5.9 The Participants Interpretation of simplified CS:GO level

A participants interpretation of what a simplified CS:GO level mean may have influenced their design and completion time, see 4.1.

5.10 Diverging Data

The way the designer maneuvers in Unreal Engine may differ from other engines or softwares which can allow the users past experiences to impact the results.

Chapter 6

Conclusions and Future Work

In this study, a new workflow regarding the creation of a CSGO level was implemented and evaluated against the already existing workflow in terms of efficiency and satisfaction. The levels requirements was to match the rule of thumbs regarding the travel time of connecting paths between the levels key elements.
In this chapter, the research questions are answered with data gathered from the experiment earlier presented the thesis. The credibility of the results are however questionable since the study only had data from 9 participants which can let a users past experiences and other factors bias the results. Also since the Hammer editor was not used in the experiment, it was not possible to create a perfect replica of the workflow used during the creation CSGO maps. But since there seems to be a sizable difference between the methods, future development of similar tools look promising.

RQ1: Which workflow is most efficient when it comes to calculate the travel time of connecting paths between key elements in a competitive team-based multiplayer fps?

Answer: In this study, the new implemented the *Timing Tool* was measured to be the most efficient workflow.

RQ2: Which workflow is by the users perspective most satisfying to work with?

Answer: The new implemented the *Timing Tool* was considered by the participants to be the most satisfying workflow.

6.1 Evaluating the *Run & Time* Workflow in the Hammer Editor

It would be interesting to see an accurate evaluation of the *Run & Time* workflow in the Hammer Editor. It's uncertain if the real workflow, in the Hammer editor, would receive better scores from the efficiency and satisfaction evaluation.

6.2 The *Timing Tool* Implemented in the Hammer Editor

Considering the scores received from this study it would be interesting to have the *Timing Tool* implemented in the Hammer Editor. If such is possible in the current version of the Hammer Editor is not certain, but if future development Counter-Strike series occur, it may be possible in a future version.

6.3 The *Timing Tool* Tested in other Genres

The scope of the research was narrowed down so that the *Timing Tool* only got tested in the bomb defusal game mode of CSGO. However, it would be interesting to evaluate the tool in other types of games and other types of genres. Especially in other multiplayer games containing asymmetrical levels that also requires levels to be balanced. The tool could also be used in single player games, to map out how far the player character is able to run a certain music score ends.

References

Adams, E. (2014). *Fundamentals of game design* (Third edition ed.). Berkeley, CA: New Riders.

Galuzin, A. (2018, February). *World of Level Design - Tutorials for Becoming the Best Level Designer and Game Environment Artist.* Retrieved 2018-04-10TZ, from `http://www.worldofleveldesign.com/`

Li, R. (2017). *Good luck have fun: the rise of eSports.* (OCLC: 1011647347)

Makin, O., & Bangay, S. (2017). Orthogonal analysis of StarCraft II for game balance. In (pp. 1–4). ACM Press. Retrieved 2018-05-07TZ, from `http://dl.acm.org/citation.cfm?doid=3014812.3014844` doi: 10.1145/3014812.3014844

McCutcheon, C., Hitchens, M., & Drachen, A. (2018). eSport vs irlSport. In A. D. Cheok, M. Inami, & T. Romão (Eds.), *Advances in Computer Entertainment Technology* (Vol. 10714, pp. 531–542). Cham: Springer International Publishing. Retrieved 2018-05-08TZ, from `http://link.springer.com/10.1007/978-3-319-76270-8_36` doi: 10.1007/978-3-319-76270-8_36

Mifsud, J. (2015, June). *Usability Metrics - A Guide To Quantify The Usability Of Any System.* Retrieved 2018-04-10TZ, from `https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/`

Newheiser, M. (2009, March). *Playing Fair: A Look at Competition in Gaming.* Retrieved 2017-06-08TZ, from `http://strangehorizons.com/non-fiction/articles/playing-fair-a-look-at-competition-in-gaming/`

Novak, J. (2012). *Game development essentials* (Third edition ed.). Clifton Park, NY: Delmar Cengage Learning.

Oxland, K. (2004). *Gameplay and design.* London ; Boston: Addison-Wesley.

Rollings, A., & Adams, E. (2003). *Andrew Rollings and Ernest Adams on game design* (1st ed ed.). Indianapolis, Ind: New Riders.

Rollings, A., & Morris, D. (2004). *Game architecture and design* (New ed ed.). Indianapolis, Ind: New Riders.

Ross, B. (2014, July). *The Visual Guide to Multiplayer Level Design.* Retrieved 2017-04-17TZ, from `http://www.gamasutra.com/blogs/BobbyRoss/20140720/221342/The_Visual_Guide_to_Multiplayer_Level_Design.php`

Shekar, S., & Karim, W. (2017). *Mastering Android game development with Unity: exploit the advanced concepts of Unity to develop intriguing, high-end Android games.* (OCLC: 964379215)

Appendix A

Supplemental Information

Hej *name*, härmed bjuds du in till att delta i ett användbarhets experiment på tisdag eller fredag, 8 och 11 maj på BTH om det är så att du uppfyller användarkravet för studien.
Användarkravet för att få göra testet är dels att ha 20 timmar eller mer total speltid på ett utav spelen i Counter-Strike serien.

Syftet med experimentet är att utvärdera vilket utav två tillvägagångssätt som är mest användarvänligt och effektivt inom skapandet av en CS:GO bomb defusal karta. Med hjälp av datan från experimentet får vi förhoppningsvis en djupare förståelse som kan förenkla skapandet av framtida kartor inom spelläget (game mode) bomb defusal.

Under experimentet kommer du få använda två olika tillvägagångssätt inom skapandet av en simpel Counter-Strike Global Offensive karta och sedan svara på en 10 korta frågor. Testet beräknas vara ca 35 minuter. Resultatet kommer att presenteras som en del i en kandidatuppsats. Datan kommer behandlas konfidentiellt.

Experimentet kommer hållas i rum J3229 Edith Clarke, direkt till höger efter att man kommer in i lärarkontorsområdet. Se bildbeskriving i länken nedan.

Anmäl dig på en ledig tid här!

https://docs.google.com/spreadsheets/d/1CGoeZl8zWtwuOe8wwTzy6wzbeL89B6Za_VTHEi8ffQE/edit?usp=sharing

Härmed bjuds du in till att delta i ett användbarhets experiment på tisdag eller fredag, 8 och 11 maj på BTH om det är så att du uppfyller användarkravet för studien.
Användarkravet för att få göra testet är dels att ha 20 timmar eller mer total speltid på ett utav spelen i Counter-Strike serien.

Syftet med experimentet är att utvärdera vilket utav två tillvägagångssätt som är mest användarvänligt och effektivt inom skapandet av en CS:GO bomb defusal karta. Med hjälp av datan från experimentet får vi förhoppningsvis en djupare förståelse som kan förenkla skapandet av framtida kartor inom spelläget (game mode) bomb defusal.

Under experimentet kommer du få använda två olika tillvägagångssätt inom skapandet av en simpel Counter-Strike Global Offensive karta och sedan svara på en 10 korta frågor. Testet beräknas vara ca 35 minuter. Resultatet kommer att presenteras som en del i en kandidatuppsats. Datan kommer behandlas konfidentiellt.

Experimentet kommer hållas i rum J3229 Edith Clarke, direkt till höger efter att man kommer in i lärarkontorsområdet. Se bildbeskriving i länken nedan.

Anmäl dig på en ledig tid här!

https://docs.google.com/spreadsheets/d/1CGoeZl8zWtwuOe8wwTzy6wzbeL89B6Za_VTHEi8ffQE/edit?usp=sharing

A.1 Consent Form

1 Usability Test Consent Form

I'm a Technical Artist student at BTH under Dr. Yan Hu. We are asking you to be in a study to help us evaluate the efficiency and satisfaction of two workflows for a bachelor degree project.

Feel free to ask any questions to us that you have before or during the test. The test should not exceed 45 minutes in length.

In this test:

- You will be asked to create a simple game-level with two different workflows in the game engine Unreal Engine 4.
- You will be asked to fill in a post-test questionnaire.
- Your screen will be recorded for purpose of later playback and gathering of data.

Participation in this usability study is voluntary. You can withdraw your consent to the experiment and stop participation at any time. Any information that is obtained will remain confidential and will be disclosed only to the researcher and the supervisor.

There are no risks involved in participating in this activity, beyond those risks experienced when sitting infront of a computer.

I have read the information above. By signing below and returning this form, I am consenting to participate in this survey/questionnaire project as designed by the below named Blekinge Tekniska Högskola student.

Participant name: _____

Signature: _____

Date: _____

Please keep a copy of this consent form for your records. If you have other questions concerning your participation in this project, please contact me at:

Student name: Jesper Eriksson
Telephone number: 070 744 79 44
Email address: jeea14@student.bth.se

Or my Blekinge Tekniska Högskola course supervisor or coordinator at:

A.2 Experiment Description

1 Experiment Description

As a participant of this test, you shall be experienced with the game Counter Strike Global Offensive (CSGO), the game mode bomb defusal. This test will focus on a simplified version of the level creation of a bomb defusal map.

In CSGO, two teams are competing against each others on a level also known as a map. The Counter-Terrorists (CT) and the Terrorists (T). In order to not have one of the two teams more preferable than the other in terms of win percentage, the levels need to be balanced. Meaning that the level has been tuned in the creation phase to prevent just that. To quicken the process of creating a balanced map, the design community have come up with rule of thumbs and guidelines regarding the timing between key areas on the level. Meaning, the time it takes for the player character to run at maximum speed from one key area to the other. The key areas in this simplified level is:

- Spawn areas:
  The location of a teams starting position.
- Bombsites:
  The area where the bomb can be planted.

And the rule of thumbs regarding the timing between the above mentioned key elements are:

- The time between CT spawn area and T spawn area: 45 seconds.
- The time between CT spawn and the bombsites: 15 seconds.
- The time between T spawn and the bombsites: 30 seconds.
- The time between the two bombsites: 20 seconds.

You as a participant will be using the game engine Unreal Engine. In that engine you will place spawn areas and bombsites in an empty level and create connecting paths between the two as fast as you possible can. Your task will be considered completed if the running-time of the paths differs less than 1 second from the corresponding rule of thumb time. Two workflows for creating paths in relation to the rule of thumbs is presented below. You will attempt creating a level using each of these workflows in turn. Immediately after finishing testing a workflow, a questionnaire consisting 10 questions with a likert scale of 1 -5 will be presented.

Workflow 1:

- Shape a level using simple geometry
- Load the level by clicking play

A.3 Pre-Assessment Questionnaire

Table 1: Level Design

	Strongly disagree				**Strongly agree**
1. Are you experienced with level design?	1	2	3	4	5

A.4 Satisfaction Questionnaire

Table 1: Workflow 1

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

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden