Examensarbete
15 högskolepoäng, grundnivå

A Study on Controllability for Automatic Terrain Generators

En studie på kontrollerbarhet för automatiska terräng generatorer

Anton Sukanya Arnoldsson

Examen: Kandidatexamen 180 hp
Handledare: Carl Magnus Olsson
Huvudämne: Datavetenskap
Program: Spelutveckling
Examinator: Carl Magnus Olsson
Datum för slutseminarium: 2017-06-01
A Study on Controllability for Automatic Terrain Generators

Anton Sukanya Arnoldsson
Comp. Science with specialization in Game Development
Malmö högskola
Malmö, Sweden
anton.arnoldsson@hotmail.com

Abstract—Procedural Content Generators (PCG) typically excel at generating a large amount of content in a short period of time. Whilst this is making PCG very applicable for the game industry, simplistic implementations of PCG lack in Usability whereas complex implementations of PCG lack in Controllability.

The purpose of this study is therefore to deepen our understanding on the correlation between Controllability and Usability in algorithmic generators that utilizes a generic and constructive approach to generate terrain in games. Furthermore the findings in this study can be used in the field of procedural terrain generators to study deterministic generators that utilize Automatic generation, from a Usability or Controllability perspective.

Keywords—Procedural Content Generation; PCG; Controllability; Usability; Designability; Expressive Range; QUIM; Quality in Use Integrated Map; GDQA; Graphical Dynamic Quality Assessment; Perlin; Worley; expressive range; expressive range method; sub characteristic; test terrain; test participant; advanced generator; perlin noise; basic generator; design science paradigm; software quality model; fractional brownian motion; appropriateness recognizability; software quality; usability characteristic; usability trait

I. INTRODUCTION

Procedural Content Generators (PCG) typically excel at generating a large amount of content in a short period of time. Whilst this is making PCG very applicable for the game industry, simplistic implementations of PCG lack in Usability whereas complex implementations of PCG lack in Controllability. We define the concept of Designability as the balance between Usability and Controllability. Previous research [15,16,17] has shown how Expressive Range [1] can be utilized to evaluate the diversity of generated content for a generator and its particular state of input variables. To test the correlation between Controllability and Usability specifically for terrain generators an application for creating and editing generators is thus needed. The application will contain common generation techniques [2,13,18,19] that the designer can use to design a generator capable of generating terrains applicable in games. By comparing Expressive Ranges and studying how the diversity changes when produced by different setup generators within the application we can evaluate the controllability of the created generators. We choose the Quality in Use Integrated Map (QUIM) model [3] for assessing the usability in our software application TGen.

The purpose of this study is to deepen our understanding on the correlation between Controllability and Usability in algorithmic generators that utilizes a generic and constructive approach. What characterizes a deterministic terrain generator with high designability, which we define as the correlation between Controllability and Usability?

We propose a study that follows the design-science paradigm [7] in which we will study, specify and measure the Usability of the application using certain aspects of the QUIM model [3] together with the Graphical Dynamic Quality Assessment (GDQA) model [8] on criterias and metrics collected using International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), ISO/IEC 9126 [4] and ISO/IEC 25010 [5] as foundation. Test participants will use the application to try and recreate a terrain by comparing their artefact to a given control artefact tweaking the inputs until they are satisfied. The Expressive Range differences of the two artefacts will then be compared to evaluate the Controllability of the application. Understanding how the Designability changes in different generator-sets gives us insight into what makes a generator great in the first place. Furthermore the findings in this study can be used in the field of procedural terrain generators to study deterministic generators that utilizes Automatic generation, from a Usability or Controllability perspective.

This paper is split up in five sections. The Introduction; a Related Research section where we present previous research
and frameworks that we use in our thesis as well as define our own framework (Designability) for the thesis; a Method section where we explain the methodology and define our method; a Data section where we present the data we have collected; a Discussion section where we discuss our findings and a Conclusion section where we draw conclusions.

II. RELATED RESEARCH

In games today PCG is being used more and more often because of the advantages that comes with it. Using PCG to create game content can be very time-effective thus cost-effective because of the power a well constructed generator possesses. PCG can also present seemingly infinite amounts of gameplay content to the player by utilizing real-time content generation to increase the replayability of a game. There is a wide variety of content generation problems and methods available within the field of PCG. In this study we will focus only on terrain generators. To further categorize the differences and similarities between the many types of generators there are in a field as broad as PCG we use the PCG taxonomy presented [14] and revised [13] by Togelius et al. In this paper we focus on PCG techniques that attain the taxonomy traits we seek. Namely the traits, Offline, Necessary, Random Seed, Generic, Deterministic, Constructive and Automatic Generation (See Table I for explanation).

Working with generators has its downsides as well. By having the computer assist the designer in the creation of content, the designer’s actions are now dependent on the Usability trait of the application that controls the generator. It’s self-evident that as the designer distributes more of the designing workload onto the computer (by designing a more advanced generator), the Controllability of the application will diminish. With a larger and more complex rule set there is now more assumptions that the computer has to make and the risk of rules contradicting or affecting each other in an unpredictable manner is increased. The generator’s design outcome runs at risk of directly conflicting with the designer’s vision of the outcome. In other words, with a more complex rule set there is a higher risk of having a low Controllability trait in a generator (because of complexity and possible contradictions between generator input variables). On the other hand with a simple rule set comes not only the chance of increased Controllability but also the lack of Usability in the application that controls the generator. Now the designer has a lot less fields to specify for the generator to work its magic. The designer has fewer ways to adjust the generator but the computer also has fewer ways to interpret the rules as a whole and furthermore a smaller risk of making the wrong assumptions. Creating a high quality generator becomes a brinkmanship between Controllability of the generator and Usability of the application that controls the generator.

I. Expressive Range

Within the field of PCG, Expressive Range [1] is a well established technique for assessing the diversity and controllability of generated content. It has been used to measure the expressivity for procedural world generators that create adaptive content using a generic method [15] as well as for stochastic generators that make use of the search based approach [16]. Expressivity tries to explain how different metrics depend on each other and tries to give a bigger picture on which values can be reached by the generator and the limitations of the generator and its generative spectrum.

Getting the Expressive Range of a generator is done in four steps: Determining appropriate metrics, Generating content, Visualizing generative space and last but not least Analyzing the impact of parameters.

1.1 Determine appropriate metrics

For evaluating the expressive range, a set of metrics should be defined. These metrics can then be measured to evaluate the range of content that the generator can create. It is important to choose metrics that are based on global properties of the generated artefacts. The metrics should ideally be comparable qualities that the end-user might notice while being exposed to the artefact. To get interesting and useful results it is important that the chosen metrics differ from the input variables of the generator. If the metrics are too similar to the input variables, the resulting change will be linear, and no longer be useful for calculating diversity. When choosing metrics it is also important that the different metrics have a vague or seemingly non-existent relationship to each other, since that will yield the actual relationship between these metrics as a result. If the different metrics are too similar or have an obvious and/or linear relation, then the measurements will be clumped together in the resulting heatmap visualization making readings of the diversity range less accurate. The amount of metrics should be at least two, note however that for visualization each metrics needs their own dimension in the heatmap, which makes visualization of four or more metrics not very reader-friendly.

1.2 Generate Content

When the metrics has been defined, the content generation can start. To create accurate measurements a lot of iteration is required to create a large amount of generated artefacts. The only change between each iteration during the Expressive Range tests is the change of seed for the generator, which is pseudo-randomly chosen for each iteration. The amount of iterations, thus the amount of generated artefacts should be enough to generalize the measurements of the metrics and produce a probability cloud shape in the heatmap. The paper on Expressive Range [1] uses 1000-10000 iterations, which seems to produce good looking heatmaps in all cases. After generating an artefact the defined metrics calculations are applied to that artefact and the measurements are stored away for later use.
1.3 Visualize generative space

To view the expressive range of all measurements gathered by the generated content in a way that gives a good overview, heatmaps are suggested. With two metrics we can express the results in a 2-dimensional heatmap with each dimension representing each metrics. Then using the grayscale spectrum we can visualize the number of iterations where the metrics values end up on their respective position on the axes.

1.4 Analyze impact of parameters

By changing the input parameters in the generator we can now visualize the effect the change has on the output artefacts by comparing different graphs with each other. Because each of the heatmaps consists of results from a large amount of generated artefacts, the individual differences between artefacts is washed out, giving us a clear indication to how changing one of the input parameters influences the final result of the output.

2. Quality in Use Integrated Map

QUIM is a framework for quantifying usability metrics in software quality models [3]. The framework proposes the Quality in Use Integrated Map (QUIM) for specifying and identifying quality components taken from different Human Computer Interfaces and Software Engineering Models. These components namely: factors, criteria, metrics and data defines the pyramid structure that makes up QUIM. It is important to remember that the different levels in the pyramid shape of QUIM are linked together, some factors depend on criterias which in turn depend on metrics which can be predicted or calculated with the help of the data variables in the QUIM pyramid. To analyze the interaction and relationship between the components in QUIM the Graphical Dynamic Quality Assessment (GDQA) [8] method is used. In this paper we will use the same framework and the same metrics variables presented in the original QUIM paper [3] which are said to be generally suitable for all software application.

2.1 QUIM components

![QUIM Diagram]

**Factors:** The following are factors that are included in QUIM
1. Effectiveness: The degree of accuracy and completeness with which the user achieves a specified task in a certain context.
2. Efficiency: The amount of resources expended in relation to the accuracy and completeness with which the user achieves a goal.
3. Satisfaction: Freedom from discomfort and positive attitude towards the use of the software product.
4. Productivity
5. Safety
6. Interoperability: The degree to which software can be used for adjustable and serviceable purposes, taking into account variations in regions, population stereotypes, languages, and cultures.
7. Accessibility: The degree to which software can be used comfortably by a wide variety of people, including those who require assistance technologies such as screen magnifiers or voice recognition.

**Criteria:** Like many others model, criteria are sub-factors. The difference is that they are measurable through a set of metrics. The following are examples of criteria included in QUIM:
1. Attractiveness: An indicator expressing the extent to which user likes the software during the operation.
2. Consistency: A characteristic of user interface that bears on the uniformity of user interface.
3. Minimal Action: The extent to which user needs to take minimal effort to achieve a specific task.
4. Minimal Memory Load: The extent to which user needs to keep minimal amount of information in mind to achieve a specified task.
5. Completeness: The extent to which the user can complete a specified task.

**Metrics:** Calculated specifically for each unique software.

**Data:** Identified specifically for each unique software.

2.2 Metrics

QUIM identifies over 100 usability metrics, some of them are functions and some are just simple countable data. The following is an example of a metric included in a QUIM case study, as stated in section 5, Case Study: Using QUIM and GDQA for Defying the ISO/IEC 9126 Model, for Usability of the QUIM article [3]:

1. Completeness of function understood: \( IUX1 = \frac{IUA1}{IUA2} \).

2.3 Data

The basic of QUIM is the data that is required to estimate metrics. The following are examples of data (variables) included in a QUIM case study, also stated in section 5, Case Study: Using QUIM and GDQA for Defying the ISO/IEC 9126 Model, for Usability of the QUIM article [3]:

1. IUA1: Number of functions understood.
2. IUA2: Total number of functions.
2.4 QUIM with GDQA

Graphical Dynamic Quality Analysis (GDQA) [8] is a method that was developed in order to graphically present the quality requirement specifications in a system. The method is based on a logical-based framework which assumes that every quality requirement can be expressed as a multi-variable function independent of each other. The steps of the GDQA method can be summarized into following three-steps:
1. Decompose each quality factors (requirement) into hierarchical level until reach their metrics.
2. Decompose each metric into hierarchical level until reach data, which are necessary for calculating these values.
3. Identify the relationships between factors and data can be an entity, a simple calculation (division, multiplication etc.) or a complex formula (prediction model) that helps to calculate related values of factors.

In the article the quality factors used in step 1 are acquired from the ISO/IEC-9126 [4] a standard for product quality in the evaluation of software quality. We will use the same quality factors except we will acquire them from the updated ISO/IEC-25010 [5].

2.5 ISO/IEC-9126

The ISO/IEC-9126 standard for evaluation of software quality [4] was replaced by ISO/IEC-25010 on March 1, 2011 [5]. The differences between ISO/IEC-25010 and its predecessor are many but in this paper we will only be focusing on the differences that concern the Usability characteristic of the standard’s definition.

There are two new sub-characteristics in ISO/IEC-25010, namely:
1. User error protection: Degree to which a system protects users against making errors.
2. Accessibility: Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

There is also name changes for two of the sub-characteristics. Understandability is renamed to Appropriateness Recognizability, and Attractiveness is renamed to User Interface Aesthetics.

3. PCG Taxonomy & Techniques

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online versus offline</td>
<td>Offline refers to the content generation taking place during the development process. In contrast online refers to content generation in real-time while the player playing the game.</td>
</tr>
<tr>
<td>Necessary versus optional</td>
<td>Necessary refers to whether or not the generated content is required to successfully complete the goal of the game.</td>
</tr>
<tr>
<td>Random seed versus parameter vectors</td>
<td>Random seed refers to generators that use a pseudo-randomly generated number to control the content creation of the generator.</td>
</tr>
<tr>
<td>Generic versus adaptive</td>
<td>Generic refers to generators that create content without taking the player's actions into account.</td>
</tr>
<tr>
<td>Stochastic versus deterministic</td>
<td>Deterministic refers to generators that can regenerate the same output content for the same input content.</td>
</tr>
<tr>
<td>Constructive versus generate-and-test</td>
<td>Constructive refers to generators that generate in one pass, without any internal quality assessment that can trigger a regeneration.</td>
</tr>
<tr>
<td>Automatic generation versus mixed authorship</td>
<td>Automatic generation refers to generators that take simple input variables from the designer or player to generate content. In contrast mixed authorship (or mixed initiative) refers to tools where the designer or player work together with the generator to create content.</td>
</tr>
</tbody>
</table>

* From PCG taxonomy presented [14] and revised [13] by Togelius et al

There are a number of recognised techniques and algorithms that will suffice to procedurally generate content according to the taxonomy traits specified, namely the traits; Offline, Necessary, Random Seed, Generic, Deterministic, Constructive and Automatic Generation (See Table I for explanation).

Both functions described below rely on deterministic algorithms to generate noise patterns [2,13,18,19]. Noise has been exceptionally useful in procedural terrain generation because the noise values can be used to describe height values in a heightmap. Many of the techniques presented in this paper were originally invented with the intent of being used in texture generation. Now these techniques are iconic methods for generating terrain elevation. In this paper we will focus on algorithms that can be used specifically to generate realistic terrain height features and characteristics.

3.1 Perlin Noise

Developed by Ken Perlin in 1983, Perlin Noise [9] is one of the most popular and most commonly used noise algorithms in the computer graphics and motion picture industry. The algorithm uses a type of gradient noise to generate value maps of smooth coherent noise.

Raw Perlin Noise is rarely used on its own. Instead it is common [2,13,18,19] that the noise is used together with a fractal function, such as the Fractional Brownian motion [11]
function. By rescaling the noise and adding it into itself using the Fractional Brownian motion function this generates the iconic Perlin noise texture that is often portrayed and also most commonly used in terrain generation and cloud texture generation. Perlin Noise can be used in arbitrary high dimensions however it should be noted that above the third dimension other algorithms will perform a lot better with similar results. Simplex Noise algorithm [12] is an example of one of these algorithms, also developed by Ken Perlin.

3.2 Worley Noise

Developed by Steven Worley in 1996, Worley Noise [10] is a common noise algorithm [2,13,18,19] that generates a quite unique noise pattern. The algorithm uses a Voronoi diagram to express the distance from points in the diagram as values in the value map. This generates the iconic Worley Noise texture that is often used to texture rocks, water and model cell objects.

4. Designability

We propose a new framework and define Designability traits in software as a balance between having both Controllability and Usability traits in the software. We base our definition and evaluation of the characteristics on accepted research from the related research. In the field of PCG Expressive Range [1,15,16,17] can be used to determine the Controllability of a generator. Generators with high Controllability can be more easily utilized by designers to generate high quality artefacts. We will use it specifically to evaluate terrain generators. In software the Usability can be measured and calculated [3]. The Usability trait directly affects how well the designer can use the generator to generate artefacts. Our framework Designability attempts to explain how these different traits correlate to influence the quality of the output artefacts by iteration of generated artefacts for Controllability, and calculated metrics using data from the study survey for Usability.

4.1 Controllability

Controllability in procedurally generated content is defined as to which degree the generator’s output content can be controlled by the generator. In other words, it defines how easy it is to control the changes of the generated output by changing the input parameters of the generator. In this definition there are several factors to consider. The designer needs to be able to control the generated content by altering the input variables. Changing the correct input variables should make the generated content have more and more similarities with the final visual that the designer aspires to. This could for example be what the designer visualizes as perfect content in the given situation. However the overall control of the generator also depends on how different generated artefacts (with the same input variables except seed) are from each other. A Generator that generates similar generated artefacts for every iteration is also a generator with higher Controllability compared to one which lack this behaviour of consistency.

For this we define two sub-characteristics, Control and Diversity to be part of our Controllability characteristic.

4.2 Control

This defines to which degree the designer can control the output content using the input variables of the generator. In a good generator, this value should be as high as possible. However it should be noted that as the size of the generator grows, from new features being added, the control diminishes. This is because with more features there are more rules that the generator must follow and the risk of rules overlapping and contradiction increases. On the other hand, with higher control the work is transferred onto the designer and less of the work is done by the generator. High Control contributes to high Controllability. We measure Control by comparing heatmaps from Expressive Range [1] testing. By comparing the Expressive Range of artefacts created in the study with the example artefacts’ Expressive Range (Generated from the example terrain in the different tests), we will be able to see what level of control each test participant had while using TGen to create generators in the different tests.

4.3 Diversity

The diversity of the generator defines how much generated artefact with the same input conditions (except for the seed) differ from each other. We measure this by looking at the generated output contents. The seed is changed between each iteration to make random number calculations differ. If the generator creates similar looking generated artefacts with every iteration that means the generator has a low diversity sub-characteristic. Low Diversity contributes to high Controllability. We measure Diversity by looking at the spread of the results in heatmaps from Expressive Range [1] testing. By comparing the spread in Expressive Ranges of artefacts created in the study with the spread in the example artefacts’ Expressive Range (Generated from the example terrain), we are able to see the levels of diversity.

4.4 Usability

Usability is the other main characteristic we define for the Designability framework. There are many models for defining Usability in software applications. We base our definition of the Usability trait with the help of the QUIM framework [3] together with the GDQA model [8] using an accepted set of usability characteristics taken from ISO/IEC-25010 [5]. We will also refrain from measuring some of the characteristics in QUIM and ISO/IEC-25010. We do this to simplify the measurements and to narrow down amount variables we need to include in our tests. The limited use of the different frameworks and models are listed below together with the reasoning behind why we limit its use by excluding some of its properties.

4.5 QUIM Factors Selection

We will focus on three factors, Effectiveness, Efficiency and Satisfaction, however we won’t be using the factors to a
bigger extent because we follow the GDQA model [8] which identify criteria and work its way down the QUIM pyramid to find the specific data variables. We still choose these three factors because they have a direct relationship with how well the generator will perform from the designer’s point of view. We exclude the factors Productivity, Safety, Internationability and Accessibility for the following reasons:

4. Productivity: Generators in the field of PCG have high productivity, one of its strong points. The minimal action of the designer (compared to the actions made by the generator) is so small it can be discarded.
5. Safety: This is mainly measured if the software that has real-world application, where a safety factor is to be expected.
6. Internationability: This is not relevant for measuring the general designability of a generator.
7. Accessibility: This is not relevant for measuring the general designability of a generator.

4.6 ISO/IEC-25010 Characteristics Selection

We will only focus on the Usability characteristic of the different characteristics in ISO/IEC-25010. Furthermore, we will only focus on two sub-characteristics, Appropriateness Recognizability (Also known as Understandability) and Learnability.

1. Appropriateness Recognizability: This will help us measure if the designer understands how the generator’s different input fields interact with the generator.
2. Learnability: This will help us measure how well the designer can use the help tools to further understand the generator.

These two sub-characteristics will be the foundation on which we define our criterias, similar to the criteria used in the case study of the QUIM paper [3].

4.7 Designability Summary

All put together, Designability will deepen our understanding on the correlation between Controllability and Usability for deterministic generators with automatic generation in the field of procedural terrain generators for games. Understanding how the Designability changes in different generator-setups gives us insight into what makes a generator great in the first place, in terms of balance between Controllability and Usability.

III. Method

In this paper we conduct a study that follows the design-science research approach [7]. The study is needed to collect information that that we use in QUIM [3] together with GDQA [8] and ISO/IEC-25010 [5] to evaluate the sub-characteristics (Usability, Controllability) of our framework Designability. The study also produces artefacts that we use in the Expressive Range model [1] to calculate different aspects of the generator. The overall method for designing, testing and evaluating our framework, we base on the design-science paradigm.

In the coming sections Data and Discussion we will respectively present and discuss the following steps:

### TABLE II. Sub-Section Layout

<table>
<thead>
<tr>
<th>Sub-Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Expressive Range findings</td>
<td>We present and discuss the data collected through the use of Expressive Range tests on the generated artefacts. This section will present and discuss data from the basic, advanced and custom test.</td>
</tr>
<tr>
<td>2. Usability findings</td>
<td>We present and discuss the data calculated through the use of QUIM [3] together with GDQA [8] and ISO/IEC-25010 [5] on data collected through the survey. This section will present and discuss data from the basic and advanced test.</td>
</tr>
</tbody>
</table>

Subsections to expect in the coming sections Data and Discussion

1. Methodological background

The design-science paradigm which includes the design-science research approach has its roots in engineering and the sciences of the artificial [6]. The original purpose of the design-science methodology was to develop technology-based solutions to important and relevant business problems. The design-science paradigm seeks out to create “what is effective”, in contrast to behavioral-science paradigm which seeks out to find “what is true” [7]. This makes it a fitting method for researching PCG and computer software effectiveness and efficiency in general.

2. Research setting

Reviewing the available related research, we design the study in sections. By introducing increasingly difficult challenges for the participants to solve, data can be partly extracted even in cases where the participants can’t solve all of the problems presented to them. The population of the study comprise of people with basic computer knowledge. A brief introduction to Unity and the TGen application is given to all of the participants. This introduction is purposely simplified to affect the study as little as possible and only includes description of the controls for Unity as well as the controls for the TGen application.

The study participant is given a ready-to-use Unity project where everything needed to perform the tests is already set up. Each test is packaged in its own Unity scene. The raw data is partly collected through the artefacts produced by the study participants. To measure the Expressive Range [1] we need to decide on two properties of the terrain that we will use as the metrics for the Expressive Range. We chose Linearity which specifies how linear the generated terrain landscape is similar to what many others [1, 15, 16, 17] have measured previously and Oceanity which specifies how much of the generated terrain that is covered in water. We choose Oceanity because we feel that for a Terrain Generator, water correctly reflect an obstacle in the terrain, making it have similar measurable effects as the Lenity metric describing level difficulty, used
in the Expressive Range paper [1].
Raw data is also extracted from the form each participant fills out in the end of the study. Specifically, the information gathered from the form will be used to determine the content of the QUIM’s Data characteristics which then can be used together with GDQA and ISO/IEC-25010 to calculate our three main focus criteria, Understandability and learnability, as well as the factors, Effectiveness, Efficiency and Satisfaction.

3. Research approach

To fully test the capabilities of different procedural content generators we develop a generator application that we call TGen. This application is developed using Unity version 5.6. When developing TGen we took the chosen PCG taxonomy in mind. We refrain from adding any extensive tool support in our application and we do not include any mixed initiative (mixed authorship) functionality in the application. We limit ourselves like this because our main goal is to research the Controllability and Usability of automated generation generators. Thus we direct any similar study on mixed initiative (mixed authorship) to the future research. TGen supports a collection of the PCG techniques [2,13,18,19] previously mentioned and gives the user the opportunity to combine these techniques in a unique fashion. The techniques that the application uses are deterministic, seed based, generic and constructive by nature. Furthermore our application has the intent to be used for offline and necessary generation. Because of how the node editor plugin that TGen uses works, the nodes can be edited and the generator can be changed without any back-end coding required.
For the study we are looking for testers that ideally have great experience with Computer Science, Game Development and Procedural content generation, however we opted to include anyone and adapted the TGen application instructions to also appeal to people without Computer Science experience. In other words, we create the application TGen with regards to a non-perfect participation group. We make sure that the application can be used by anyone with average computer and designing skill. We do this to minimize the possible negative impact that unsuitable test subjects could bring.
In the study the participants use TGen together with test scenes specifically created for the purpose of the study to produce their artefacts. The study consist of three tests:

<table>
<thead>
<tr>
<th>Study Tests</th>
<th>Description</th>
</tr>
</thead>
</table>
| T1. Basic Generator | Targets the low-end setup of the generator size spectrum. Used as a case where the generator is ‘too simple’.
| T2. Advanced | Targets the high-end setup of the generator size spectrum. Used as a case where the generator is ‘too advanced’.

<table>
<thead>
<tr>
<th>Generator</th>
<th>Description</th>
</tr>
</thead>
</table>
| T3. Custom Generator | Targets the participants preferred generator size of the generator size spectrum. Used as a control case.

3.1 Test 1. Basic generator

User is presented with a basic generator (with only one node) linked to a test terrain. User is also presented with a black-box example terrain (See Figure 1) that has previously been produced by the basic generator with its input parameters in a certain setup. It is up to the tester to use the basic generator to change the test terrain so that the test terrain becomes as similar as possible to the black-box example terrain. The artefact produced is used in the Expressive Range method.

![Figure 1](image1)

Figure 1. The example terrain used in T1 Basic Generator test.

3.2 Test 2. Advanced generator

User is presented with an advanced generator (with many nodes) linked to a test terrain. User is also presented with a black-box example terrain (See Figure 2) that has previously been produced by the advanced generator with its input parameters in a certain setup. It is up to the tester to use the advanced generator to change the test terrain so that the test terrain becomes as similar as possible to the black-box example terrain. The artefact produced is used in the Expressive Range method.

![Figure 2](image2)

Figure 2. The example terrain used in T2 Advanced Generator test.
3.3 Test 3. Custom generator

User is presented with an empty generator (no generator nodes) linked to a test terrain. User is free to test the limits of the application by creating their own generator. The artefact produced is used as a control result the Expressive Range method to ensure expressive reach of the TGen application.

3.4 Survey

In connection with the study the test participants also fill out a form. The questions in the form are aimed to produce data for calculating ISO/IEC-25010’s two sub-characteristics Appropriateness Recognizability (Also known as Understandability) and Learnability according to the GDQA model [8]. Some of the QUIM Data variables we extract from the form, others are determined simply from the way the TGen application and the tests are set up (See Table IV).

TABLE IV. DATA VARIABLES

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IUA1</td>
<td>Number of functions understood. (Extracted from Survey)</td>
</tr>
<tr>
<td>IUB1</td>
<td>Total number of functions.</td>
</tr>
<tr>
<td>IUA2</td>
<td>Number of functions identified by the user. (Extracted from Survey)</td>
</tr>
<tr>
<td>IUB2</td>
<td>Total number of actual functions.</td>
</tr>
<tr>
<td>IUA3</td>
<td>Number of interface functions whose purpose is correctly described by the user. (Extracted from Survey)</td>
</tr>
<tr>
<td>IUB3</td>
<td>Number of functions available from the interface.</td>
</tr>
<tr>
<td>ILA1</td>
<td>Number of tasks for which help is available.</td>
</tr>
<tr>
<td>ILB1</td>
<td>Number of tasks tested. (Extracted from Survey)</td>
</tr>
<tr>
<td>ILA2</td>
<td>Number of tasks successfully completed after accessing help. (Extracted from Survey)</td>
</tr>
<tr>
<td>ILB2</td>
<td>Number of tasks tested. (Extracted from Survey)</td>
</tr>
</tbody>
</table>

3.5 Adaptions

The Expressive Range method [1] was originally used on a one-dimensional set of data in two-dimensional space. This means we have to adapt the method to use chosen metrics Linearity and Oceanity. We adapt the Expressive Range method to evaluate Unity Terrain Objects represented by two-dimensional sets of data in three-dimensional space. By calculating the coefficient of determination value for every strip of terrain along the X- and Z-axis respectively we get the R2 values needed for the adaption of the Linearity calculations. This means that for a 128x128 sized terrain, there are 256 R2 values calculated, 128 times for the X side of the terrain, and 128 times for the Z side of the terrain. Linear regression quickly becomes heavy on larger terrains and an external application (MathLab) is used to speed up the calculation time. The average of all the R2 values is then used as the Linearity value, which in terms means that terrains with a higher Linearity value has the tendency to follow a straight line much better. The Oceanity metrics is calculated simply by measuring a percentage of terrain area that is covered in water. Water is created stationary on a point 8% of the maximum height above the lowest point on the terrain. This means that higher Oceanity values correspond to larger percentages of the terrain being covered by water.

As already mentioned in our definition of the Designability framework, we adapt QUIM by only focusing on the factors within QUIM that directly relates to the Usability aspects of the generator. Likewise, the use of ISO/IEC-25010 is adapted to only focus on characteristics and sub-characteristics that directly relates to the Usability aspects of the generator.

4. Data collection and analysis

The test subjects sends us the artefacts they create. The artefacts are files describing the setup of the Node Canvas for our Node Editor within TGen. Using the artefacts we can recreate the generator setup and apply the Expressive Range method on the generator. We do this to spare the study participants from using their computer power and time to go through the Expressive Range tests, which takes a lot of time to complete. Previous use of the expressive range method have used gray scale heatmaps [1] to display the quantity of generated artefacts for each value of the different metrics used in the Expressive Range method. TGen has built-in functionality to export and import text files containing generated metrics values, (that can then be imported into external software, like MathLab). TGen also supports image export and can generate heatmap images for each Node Canvas provided. We can then analyse them individually as well as create a heatmap representing the average of all the test participants results. By comparing the heatmaps given by different tests we gain insight on the Controllability characteristic of our Designability framework.

Through the collected data we create 5 heatmaps (See Figure 3-7 or Table V) each with axis values 0.0 to 1.0 for the given metrics, (Linearity and Oceanity). The heatmap will have a grayscale color depth of 5, which means that it can display values between 0 and 4+ for how many iterations end up on a specific x,y location in the heatmap. The Example Generators are iterated 40000 times while the participant’s generators are iterated 4000 times per participant. The Expressive Ranges of the participants are then added together to form a sets of 40000 generated iteration heatmap. The two heatmaps, the one from the example generator and the 10 added together from the participants are then equally strong in terms of the amount of iterations (40000).
TABLE V. Expressive Range Heatmaps

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Basic Generator</td>
<td>This represents the average of all artefacts created by the basic example terrain setup. I.e. this is the 'correct' terrain the test participants aspire to simulate.</td>
</tr>
<tr>
<td>Participant’s Basic Generator</td>
<td>This represents the overall average of all artefacts created by the participants for the basic generator test.</td>
</tr>
<tr>
<td>Example Advanced Generator</td>
<td>This represents the average of all artefacts created by the advanced example terrain setup. I.e. this is the 'correct' terrain the test participants aspire to simulate.</td>
</tr>
<tr>
<td>Participant’s Advanced Generator</td>
<td>This represents the overall average of all artefacts created by the participants for the advanced generator test.</td>
</tr>
<tr>
<td>Participant’s Custom Generator</td>
<td>This represents the overall average of all artefacts created by the participants for the advanced generator test.</td>
</tr>
</tbody>
</table>

5. Limitations

For the Expressive Range [1] to give effective results it is important to use a consistent scale for all of the tests. With an alternating scale it takes many more iterations to pinpoint the generator to the same expressivity. This creates limitations in the Terrain sizes that can be implemented. To minimize the negative impact of this we collect the Node Canvas and set the scale directly for the Expressive Range tests. This means that the user can generate quickly on a small segment size rather than the full size which also comes with a hefty generation time.

In a small study it’s harder to get accurate results. To minimize the negative effects of this, we have tried getting as many test participants as possible for our study. Using a limited set of participants is still valid as the general intent of this study is to deepen our understanding on the correlation between Controllability and Usability.

To ensure that our application TGen can deliver terrains of different types, and that the generator can reach over a wide area in the Expressive Range heatmaps produced, we added a custom test that will act as a control test to establish the limits of what TGen can produce given the right input values.

In this paper we focus on a small part of the many types of generators within the field of PCG. Determined by the PCG taxonomy [13,14] we aim our study specifically at deterministic generators that make use of automatic generation. Other implementations including the uses of mixed authorship (mixed initiative) and PCG tools we direct to future research. We do this because we feel that the Usability and Controllability traits of generators that make use of mixed authorship might be significantly different from the generators that make use of automated generation.

We limit our application TGen to only work with generating terrain, thus we are also limiting our study to focus on Terrain Generators. Measuring the correlation between Controllability and Usability for other types of generators within the field of PCG, including texture and model generation, we direct to future research.

IV. Data

Each of the heatmap pictures are of 1000x1000 pixel size. Every time an artefact is generated and goes through the use of Expressive Range according to the chosen metrics one pixel has its value changed on the grayscale. The pictures thus show us a probability cloud where white indicates a greater amount of artefacts ending up with the values described on the X- and Y-axes. For the heatmaps extracted from the study participants, every participant’s canvas was used to generate 4000 artefacts that were then studied and added together. The example generators iterated 4000 for each test participant, thus 40000 iterations were added together in every picture.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Metrics Description</th>
<th>Metrics (Xn=An/Bn)</th>
<th>Numbr of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understandability</td>
<td>Completeness of function understood</td>
<td>IXU1=IXA1/IXB1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Evident Functions</td>
<td>IXU2=IXA2/IXB2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Function Understandability</td>
<td>IXU3=IXA3/IXB3</td>
<td>2</td>
</tr>
<tr>
<td>Learnability</td>
<td>Ease of use with help</td>
<td>ILX1=ILA1/ILB1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Effectiveness of help</td>
<td>ILX2=ILA2/ILB2</td>
<td>2</td>
</tr>
</tbody>
</table>

* A Selection of Usability metrics from ISO-25010
1. Expressive Range findings

Figure 3. Expressive range of the example terrain used in T1 Basic Generator test. The probability cloud is positioned to the higher end of Oceanity and the lower end of Linearity. See Figure 1. for example of what a generated terrain would look like.

Figure 4. Expressive range findings of the generators created by the test participants in T1 Basic Generator test. The probability cloud is seemingly identical to Figure 3. which indicates high Controllability because the test participants could simulate the expressivity of the provided example terrain.

Figure 5. Expressive range of the example terrain used in T2 Advanced Generator test. The probability cloud is positioned to the lower-mid end of Oceanity and the lower end of Linearity. See Figure 2. for example of what a generated terrain would look like.

Figure 6. Expressive range findings of the generators created by the test participants in T2 Advanced Generator test. The probability cloud is very spread out and far from the original example (Figure 5), this indicates a lower Controllability because the test participants were unable to use the generator to get to the same expressivity as the example terrain.
2. Usability findings

All the values used as our $An$ metric for calculating $Xn$ in our GDQA [8] method $Xn=An/Bn$ comes from the survey all participants filled in. $An$ is an averaged value between all 10 participants. $Bn$ is evident from the different test setups. (See Table IV for more information). The Basic Generator had 4 functions (Table VIII), the Advanced Generator had 32 functions (Table IX).

![Custom Test Result Generator 40000 Iterations 256 Resolution](image)

Figure 7. Expressive range findings of the generators created by the test participants in T3 Custom Generator test. The resulting probability clouds are spread out and even separated from each other in some cases, this is because the test participants chose themselves how to construct the custom generator.

TABLE VII.  STUDY GROUP NUMBERS

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of participants</td>
<td>10</td>
</tr>
<tr>
<td>Number of iterations per Expressive Range Test</td>
<td>4000</td>
</tr>
<tr>
<td>Number of Expressive Range Tests per participant</td>
<td>5</td>
</tr>
<tr>
<td>Number of iterations in each heatmap</td>
<td>40000</td>
</tr>
</tbody>
</table>

$h$ Values from the test study group.

TABLE VIII.  BASIC ISO-25010 SELECTION

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Metrics Description</th>
<th>Metrics ($Xn=An/Bn$)</th>
<th>Result (0.00-1.00)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completeness of function understood</td>
<td>0.225=0.9/4</td>
<td>0.225</td>
<td></td>
</tr>
<tr>
<td>Evident Functions</td>
<td>0.675=2.7/4</td>
<td>0.675</td>
<td></td>
</tr>
<tr>
<td>Function Understandability</td>
<td>0.55=2.2/4</td>
<td>0.55</td>
<td></td>
</tr>
<tr>
<td>Ease of use with help</td>
<td>1.0=4/4</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Effectiveness of help</td>
<td>0.35=1.4/4</td>
<td>0.35</td>
<td></td>
</tr>
</tbody>
</table>

$h$. Results from the Selection of Usability metrics from ISO-25010 for the Basic Generator

TABLE IX.  ADVANCED ISO-25010 SELECTION

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Metrics Description</th>
<th>Metrics ($Xn=An/Bn$)</th>
<th>Number of Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completeness of function understood</td>
<td>0.452=14.45/32</td>
<td>0.452~</td>
<td></td>
</tr>
<tr>
<td>Evident Functions</td>
<td>0.766=24.5/32</td>
<td>0.766~</td>
<td></td>
</tr>
<tr>
<td>Function Understandability</td>
<td>0.613=19.6/32</td>
<td>0.613~</td>
<td></td>
</tr>
<tr>
<td>Ease of use with help</td>
<td>1.0=32/32</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Effectiveness of help</td>
<td>0.39=12.4/32</td>
<td>0.39~</td>
<td></td>
</tr>
</tbody>
</table>

$i$. Results from the Selection of Usability metrics from ISO-25010 for the Advanced Generator

V. DISCUSSION

In this section we will be talking about the findings from the Expressive Range tests, as well as the findings from the Survey. We will discuss the correlation and impact that the results have on our framework Designability and further the impact our research have on terrain generation that make use of automatic generation. By looking at the results from the Expressive Range, we will evaluate how the Diversity and the Control sub-characteristics of our Controllability characteristic relate to the study test results. By looking at the results from the ISO-25010 Selection table calculated using QUIM [3] together with GDQA [8], we will evaluate how the Appropriateness Recognizability (Also known as Understandability) and Learnability sub-characteristics of our Usability characteristic relate to the study test results.

1. Expressive Range findings

The basic generator test results Expressive Range from the study participants (Figure 4) looks seemingly identical to the basic generator example Expressive Range (Figure 3) created from the Basic Example Terrain artefact (Figure 1) that functioned as a black-box generator for the test participants during the study. Because of the few input variables the basic generator possesses, the test participants have more control.
over the changes the generator makes to the output artefact. This means that the test participants could simulate the example terrain (Figure 1) with great success. Furthermore this shows that the Controllability of the basic generator is substantial. This shows that simplistic terrain generators that make use of automatic generation have substantial Controllability.

The advanced generator test results Expressive Range extracted from the study participants (Figure 6) is significantly different compared to the advanced generator example Expressive Range (Figure 5) created from the Advanced Example Terrain artefact (Figure 2) that functioned as a black-box generator for the test participants during the study. We can see that the overall diversity of the Advanced Example Terrain (Figure 2) is lower compared to that of the Basic Example Terrain (Figure 1). This can be determined from how the values in the advanced generator example Expressive Range (Figure 5) are more clumped together when compared to the values of the basic generator example Expressive Range (Figure 3) which are more spread out. In practice however we see that this small decrease in Diversity (thus increase in Controllability) between the basic generator example Expressive Range (Figure 3) and the advanced generator example Expressive Range (Figure 5) is unsubstantial compared to the much greater lack of Control (thus decrease in Controllability) when comparing the advanced generator test results Expressive Range (Figure 6) with the basic generator test results Expressive Range (Figure 4), which are both extracted from artefacts created by the test participants of the study. Furthermore this shows that the even though the Advanced Generator has a lower Diversity, the increase in Controllability from that fact is easily overtaken by the lack of Control in the Advanced Generator, which contributes to an overall lack of Controllability for more advanced terrain generators.

The custom generator test results Expressive Range (Figure 7) refers to the Expressive Range of the custom generators that each participant had to create at the end of the study. The gathered data is used to ensure and view the full spectrum of the applications current expressivity. By averaging out different setups of generators, we see that the Custom Expressive Range wander the spectrum and that the concentration of iterations for a single spot is less at any given point. In other words, because the Custom Generator test results show a more spread out Diversity we can ensure that the TGen application itself was not a bottleneck for the expressivity findings in the Basic and Advanced Generator test results.

2. Usability findings

By viewing the results from our ISO-25010 Selection table we can see that the test participants could successfully identify how the function worked behind the scenery 22.5% of the time for Basic Generator and 45.2% of the time for the Advanced Generator. We extract this from the Completeness of function understood metrics that was collected through the survey. This means that the test participants understood the underlying concept of generator functions about twice as often for the advanced generator, when compared to the basic generator. Less distinctly we see that the test participants found that the given functions were evident 67.5% of the time for the Basic Generator compared to 76.6% of the time for the Advanced Generator. This means that for 76.6% of the functions in the Advanced Generator the participants could notice how the function changed something, possibly without understanding how the change worked and possibly without predicting how changing the value might change the resulting artefact. Function Understandability refers to functions where the test participant recognised a pattern or a shape of how the given function changes the generator output (without the need to explain how the value actually is used by the generator). We see that the Function Understandability of the Basic Generator is 55% compared to 61.3% for the Advanced Generator.

For the Learnability tests it should be noted that the tests were created to force the test participants to make use of the given help. We do this to simplify our study. We can do this without significant impact to the study because we seek to understand how well Learnability is affected rather than how often it is used. Forcing this ensures that all of the help tools of the TGen application are used during the study and also minimizes the amount of tests we have to conduct. This is also why the Ease of use with help metric is a 100%. It seems however that the difference of the Effectiveness of the help metric between the two generators is insignificantly low.

Furthermore we conclude that the Understandability trait of the Usability characteristic does show progress for more advanced setup terrain generators. We also conclude that for more advanced terrain generators both the general understanding of the overall function as well as deeper understanding of the underlying functionality is increased.

3. Summary of findings

From the generated artefacts and its collective Expressive Range [1] as well as the gathered data from the survey and its collective Usability findings from the GDQA method [8] we conclude that there is a correlation between Controllability and Usability for deterministic terrain generators that make use of automatic generation. We recognise that in the correlation between Controllability and Usability in our framework Designability, the Controllability trait is more easily disturbed. In other words, changes and additions to an automatic terrain generator will affect the Controllability trait significantly more than it will affect the Usability trait of the generator. We also conclude that even though more advanced generators have a lower Diversity, the increase in Controllability from this fact is easily overtaken by the lack of Control in the advanced generator, which contributes to an
overall lack of Controllability. We acknowledge that the Learnability sub-characteristic for the Usability trait has an insignificant impact on the Usability difference between simplistic and advanced implementations of automatic terrain generators.

VI. CONCLUSION

This paper set out to deepen our understanding on the correlation between Controllability and Usability for deterministic, generic, constructive terrain generators that make use of the automatic generation approach. Through the use of our framework Designability, that relies on previous research from Expressive Range research [1,15,16,17] and previous research on Usability traits [3,8,5], we found that there is a correlation between Controllability and Usability for deterministic terrain generators that make use of automatic generation. Our research shows that the Controllability trait is the more easily disturbed of the characteristics in the Designability framework. In this paper we focused on said correlation specifically for terrain generators. Our findings are likely to be of importance for future research on terrain generation as well as research on Controllability for automatic generation generators. We suggest more extensive testing to complement our current research because of the limitations of our study group size.

In terms of future research, we suggest similar research in areas that we did not touch upon. For example on generators that make use of the search based approach. Other aspects that could be of relevance to explore are the correlation between Controllability and Usability for generators that make use of the mixed initiative (mixed authorship) approach.

References