A Comparative Study of Representations for Procedurally Generated Structures in Games

Oscar Pleininger & David Dahl

Research field: Computer science
Bachelor Thesis
15 credits
Date for seminar: 2018-08-30
Supervisor: Olle Lindeberg

ABSTRACT

In this paper we have compared and evaluated two different representations used in search based procedural content generation (PCG). The comparison was based on the differences in performance, quality of the generated content and the complexity of the final artifacts. This was accomplished by creating two artifacts, each of which used one of the representations in combination with a genetic algorithm. This was followed up with individual testing sessions in which 21 test subjects participated. The evaluated results were then presented in a manner of relevance for both search based PCG as a whole, and for further exploration within the area of representations used in this field.

1. INTRODUCTION

The demands and expectations of what a game has to offer has dramatically increased over the last couple of decades as technology has advanced and gameworlds has gotten more and more immersive. Jesper Juul (2012) delves deeper into this discussion with two of the largest console platforms for games, Xbox and Playstation, he discussed the emerge of an "HD-era" in which the consumers demands for experiences is what drives the development forward. With this in mind it is not hard to imagine why PCG is so widely used in game development processes. The demand for good quality content that fits into today's large game worlds simply cannot be met by crafting the content manually. Therefore it is very beneficial and important to further research the field of PCG in order to streamline the way it is being used. There are of course many different aspects that affect the performance of content generation, for example which genetic algorithm that is used or which way the content to be generated is represented.

This paper is going to focus on the comparison and evaluation of two representations used to generate content for a 2D game world. After the testing and evaluation process of those representations are finished we hope to have contributed with some indicative conclusions regarding what benefits and disadvantages that are present depending on the choice of representation.

1.1 Procedural content generation

PCG is a field in computer science that refers to content being generated by one or more algorithms, thus removing the need for most if not all human interaction in the designing process of game content. Search based PCG can use a set of different representations for the content it is supposed to generate. These representations range from more direct ones, for example level maps, to more indirect ones like a random number seed. A complete list of representations commonly found in PCG has been compiled by Noor Shaker et.al (2016).

There has been a couple of previous case studies on the PCG subject conducted by Julian Togelius and Steve Dahlskog (2012) where they used design patterns as a representation of levels in the game Super Mario Bros. The goal with their study was to show that it would be possible to combine design patterns with PCG and generate new levels based on the players actions in the previous ones, among other ways of generating varied content. However, they approach the subject of level generation using only design patterns and makes no comparisons between this representation and others which is what we will be focusing on in this paper.

Another study written by Luigi Cardamone et.al (2011) was devoted to comparing four different representations that was used to evolve maps for a first person shooter game. However, since their study was based on generating maps in a 3D environment those representations that were used would differ a lot from the ones we will be using for our generation of 2D game content.

1.2 A search based approach

When using search based PCG the goal is to generate content that has high enough quality in a efficient manner. Georgios N. Yannakakis, and Julian Togelius (2011) states that using different types of representations for the genotype of the content that is generated will give the genetic algorithm different conditions which will affect how its search space is defined. Depending on the representation that is used the search space will either be narrowed down or broadened which in turn limits the amount of solutions to the problem that the algorithm has to deal with.

We intend to implement a search-based approach for creating parts of a gameworld for the first generation of Pokémon games. Pokémon is a role-playing game where you play as a Pokémon trainer and catch creatures called Pokémon. You battle other Pokémon trainers, and your goal is to defeat eight specific trainers referred to as gym leaders, allowing you to face of against four other trainers, referred to as the Elite Four, and be announced as the Pokémon Champion.

We will be using two different representations for the content that we want to generate while maintaining a similar structure for the generative algorithm. In the evaluation process of these representations we will take the quality of the generated content into consideration in relation to its performance. We will focus on generating cities where structures such as buildings, roads and other objects in the surrounding environment are distributed in such a way that it looks natural.

The reason for choosing Pokémon games was because they are a widely known 2D tile-based game series with a lot of different assets to work with in terms of object and structures present in the gameworld. This creates the opportunity to, to an extent, choose how many of

those assets that will end up being used for a particular project. All of which makes Pokémon a good environment to start with since you are able to limit the complexity of your project or expand it depending on your ambitions.

For the evaluation process of the content quality there will be a series of user tests where people will get a chance to give their opinion on whether or not the generated content could fit into the original gameworld. Our goal is to have people user testing that are aware of how the gameworlds in Pokémon games usually were designed. The results conducted from these test will then be used to discuss and evaluate our implementation of our chosen representations.

2. RELATED RESEARCH

2.1 Common representations used in PCG

There are some representations that are commonly used in search based PCG, one of which being design patterns which was used by Steve Dahlskog and Julian Togelius (2012). Other common representations are mentioned by Julian Togelius et.al (2011) in which they list different types of representations by their correlation to the phenotype of the generated content, more specifically how directly the genotype represents the phenotype. The most direct representation is when each part of the genotype represents one cell of the phenotype, e.g. grass, path or building tiles in a Pokémon world. This makes it so that the search space contains every possible combination of solutions to the given problem. This approach might not be suitable to use depending on how large the city that is going to be generated is, since it would take the algorithm far too long to find a good enough solution if the length of the genotype were to become too long.

More indirect representations for a genotype would be to either have a list of properties and their positions in the city, or to have a selection of patterns that are known to exist in a city and put these in a list together with their positions. These forms of representations have the property of containing less elements in their respective lists than a representation where the genotype is directly connected to the phenotype.

In the paper published by Julian Togelius et.al (2011) they discuss various issues with different types of representations and what to keep in mind when choosing a representation for your specific project. An example of what to look out for is that representations should have a high “locality”, meaning that a small change in the genotype most of the time should result in a small change in the phenotype, as well as have a small effect on the fitness value. They also bring up the issue with dimensionality, pointing towards the problem of having a representation that is not able to correctly represent the content, while at the same time not having a representation that has far too many possible solutions for the problem.

2.2 Common methods used in PCG

In the study conducted by Mark Hendrikx et.al (2013) they bring up different methods in PCG that can be used to generate specific parts of a gameworld. Specifically they bring up six types of generic game content that one would want to have in a game, which are: textures, sound, vegetation, buildings, behaviour, as well as fire, water, stone and clouds which are all combined into one type of content. They also bring up the relevance of generating game space, which they divide into two separate types, indoor maps and outdoor maps. The distinction between indoor and outdoor maps is defined by declaring that outdoor maps in most cases have continuous themes, which is uncommon for indoor maps since they are much more likely to have interrelated or even one of a kind artifacts.

2.3 Additional research

After their research on design patterns in PCG, Steve Dahlskog and Julian Togelius (2014) published another study on the subject where they used a combination of something called *micro* and *meso-patterns* in order to represent levels in the game Super Mario Bros. They defined their micro-patterns as slices of the original game levels, whereas meso-patterns were defined as abstractions of design patterns that was commonly found throughout the gameworld. With this representation, any given level generated would consist of a sequence of micro-patterns. However, the evaluation of the levels would be based on the amount of meso-patterns that were found in each level.

In this paper the authors also bring up an interesting issue where search based PCG often has to deal with a tradeoff between variation in the generated content and the control over what is being generated. This is something that we will have to take into consideration when constructing our representations and the way they are evaluated since this can heavily affect the overall performance.

Luigi Cardamone et.al (2011) did a comparative study where their goal was to determine which of four different representations that would produce the best maps for a first person shooter game called Cube 2. They used representations ranging from direct to indirect and in order to measure how good a map was they used the amount of time that a player could stay alive in battle as the fitness value for their evaluation function. Their reasoning for using combat time as the fitness determinator was that if a player can survive a long time after being confronted in battle, then that means that the map provides possibilities to escape, hide, find health or weapons or other ways for the player to survive. In order to evolve a wide variety of maps they then implemented a simple genetic algorithm that used single point crossover and mutations.

The four representations that were used in this study were called Grid, All-White, All-Black and Random-Digger. Grid was the most direct representation and the ones after were increasingly less direct. The Grid represented the maps as a grid of walls where each wall could be removed to create rooms and corridors. With the All-White version the maps started

out empty and content was then gradually added while the All-Black version was the complete opposite, maps would start out filled with content which would then gradually be removed. The random digger also started out with maps filled with content, just as the All-Black, and then the digger would clear all the cells that it had visited.

This study is very closely related to ours since we intend to compare different representations used for generating content for games using a search based approach. Although their study was focused solely on First Person Shooter games, the approach of generating content with genetic algorithms and finding good solutions using a fitness function will be part of our process as well, which makes this study a good reference for our research.

For the use of PCG in Pokémon games there exist a study written by Antonios Liapis (2018) which brings up the use of evolutionary algorithms as a way to get the image information from current Pokémon sprites and use that to determine the relationship between a Pokémon type and its appearance. However, for Pokémon games specifically, there appears to not have been any previous research done with the intention to compare different representations used when creating Pokémon environments.

2.4 Research question

The purpose of this study is to evaluate two different search-based representations and their potential use for generating cities for the first generation of Pokémon games. The evaluation will be based on performance, realism and complexity of the artifacts, and will be limited to only two representations.

The representations that we will be evaluating are:

- Building representation - Building structures are generated genetically while other structures are generated by an algorithm.
- Pattern representation - Every structure is generated genetically.

3. METHOD

3.1 Design research

For this study we have chosen to follow the Design Science Research Methodology (DSRM) Peffers et al (2008). This was the most fitting since DSRM focuses on research where an artifact is created and tested. It also focuses on the iterative process of building the artifact, where each step in the method can be redone as many times as necessary. These are the six steps involved in DSRM, however, it is not necessary to do the steps in this exact order.

1. Defining the problem
2. Define the goals
3. Design and develop an artifact

4. Demonstrate the artifact
5. Evaluation of the artifact
6. Communication for future work

3.2 Practical execution of the method

1 Defining the problem

The idea behind this study was to compare solutions to a search based PCG problem where a genetic algorithm would use two different types of representations in order to find variations of cities in the 2D tile-based game Pokémon. These solutions would then be evaluated based on their overall performance.The evaluation would take into consideration not only how fast the content was generated, but also if the result was good enough in comparison to the original game content. The idea behind using the original maps as references for quality was that by having the generated maps follow the "rules" of the original content, they would be more likely to fit into the Pokémon universe, which would then be used as a measurement of how good the generated maps were. The generated content produced by each solution would then be compared to each other in order to consider which would be most beneficial to use. Lastly, we also take into consideration the difference in complexity of the implementations, as this would affect the usefulness of the solutions.

2 Defining the goals

Our main goals of this study was to be able to compare two different representations used in search based PCG. In order to be able to perform these comparisons we had to have two working prototypes, each using one of the two representations. These prototypes also had to be able to generate the specific content in such a way that it could just as well have been taken from the original game. On top of that, the representations had to not be too different from each other in order to keep the comparison results relevant. In order to be able to test the generated content, the prototype would also have to be able to present its generated content in such a way that the test subject could not tell much difference from the original game.

3 Designing the artifact

In order to compare and evaluate the two types of representations used in search based PCG this paper will focus on we developed two separate artifacts in MonoGame (http://www.monogame.net) using C#, that each uses one of our representations. MonoGame is an open-source and cross-platform development environment which implements the Microsoft XNA 4 application programming interface, and allowed us an easy way to import PNG files into our program and use them to create our sprites.

4 Demonstration of the artifacts

For the demonstration of the artifacts a series of tests were conducted where people that were familiar with the original Pokémon game got to give their opinion as to which of the

generated cities that they thought were most similar to those from the Pokémon world. A more detailed explanation of our data collection process can be found in section 3.3.

5 Evaluation of the artifacts

The evaluation of the quality of the generated content was based on the results from the user tests and their feedback, while the observations of how long it would take the content to generate was the base which was used to evaluate their efficiency. Lastly, the evaluation of complexity between the artifacts was based on the amount of objects and rules that each algorithm had to take in consideration in order to produce content with a high enough quality.

6 Communication for future work

In the last step, we documented the solution as in this study to share the acquired knowledge.

3.3 Data collection

Comparison of efficiency

The measurement used for efficiency was based on a combination of two factors, the amount of generations (i) needed in order to generate maps that were of high enough quality and how much time it would take to reach the i'th generation.

Comparison of complexity

When we started to implement our prototypes for this research we had limited insight into the field of PCG. This had the effect that our first implementation took several weeks to finalize and get working as intended. What we then discovered when we went over to our pattern prototype was that both of our solutions were built up from a common ground. This meant that when it was time to build the pattern prototype we already had the building blocks needed to get started. In turn, this lead to a much faster development process for its basic functionality. What separated the two prototypes however, was the process of implementing the evaluation functions.

Comparison of quality

For the purpose of demonstrating the prototypes a testbed was constructed in which the test subjects were able to easily navigate between a set of pre-generated maps collected from both of the prototypes. The process of picking out which maps that were going to be shown was done by generating ten maps from each prototype. Then we chose five maps from each prototype that we deemed had the highest quality and showed them to the test subjects.

The testing sessions were arranged so that the test subjects would sit down and navigate the player character around in each map. Then the test subject would give each map a rating from one to five based on how well they thought the maps would fit in the Pokémon universe, one meaning not at all and five meaning it definitely would. After the ratings were set the test subjects were asked why they gave the ratings that they did.

The selection of test subjects was based on their knowledge of the early Pokémon games. People that were going to test the prototypes had to have played at least one of the three original games and be somewhat familiar with it. According to Fullerton (2014) it is important to test with people from your target audience, this way the feedback you get from the test will be much more relevant than if the testers consists of people who are not attracted to the type of game you are testing for.

4 IMPLEMENTATION PROCESS

4.1 Genotypes and phenotypes

The original game

Cities in Pokémon worlds are made up of a wide variety of structures, all of which have three different properties; visual appearance, whether the structure is solid or walkable and its position. The type of structures that the prototypes in this study are able to generate include; buildings, fences, patches of grass and flowers, roads, forests and signs.

Building representation

In the first representation the genotype consisted of a list of buildings and their positions in the city. This meant that the genetic algorithm only affected building placement and the amount of buildings in the generated cities. The remaining structures were then generated with a separate algorithm, an algorithm which had its own set of rules for how and in which order it should place the other structures. The order in which the algorithm placed out different structures were; roads, fences, forests, signs, flowers and grass. With all of these structures in place the phenotype would then be finalized. The choice of having the structures laid out in this order was made based on the observation of the original Pokémon cities. In those cities the roads and fences had the most recurring placements in relation to the buildings, while on the other hand, structures like flowers and grass were more likely to be placed in relation to roads and fences. However, since flowers and grass only differ from the roads in their visual appearance, it was deemed that the solid structure types made up of forests and signs should have a higher priority since they would affect the players ability to move around in the city.

Pattern representation

For the second prototype the genotype was made up of a list of patterns along with their properties and positions. Every type of object that could be generated into a city was represented as a pattern, some of which would be standalone, predefined patterns such as buildings and fences, while other would be different size variations of the same micro-pattern (see figure 4). This meant that the genetic algorithm for the second prototype was be able to generate the whole phenotype without any separate algorithm.

4.2 First prototype - Buildings

The first thing that was implemented into the artifact was a way to generate buildings into the cities. This would act as the base for the evaluation of the content produced by the building representation.

This representation was made up of a specific genotype which consisted of two types of terrain, empty tiles which the player was able to walk on and buildings which were solid structures made up of specific types of tiles. The buildings could vary in height all the way from two tiles up to four, but the width was always four tiles. Since there was no variations in width of the buildings the representation did not mimic every aspect of the original Pokémon cities.

The genetic algorithm will only take the placements, alignments and amounts of buildings that were available in the representations into consideration when it later would evaluate the generated maps. This would leave out other obscuring structures at this point in development. Other structures would include objects like fences, forests and signs, among others which would also typically be represented in a Pokémon city. Those elements would then be generated separately with a separate algorithm after the buildings were in place.

The next step was to implement an algorithm capable of combining the generated maps, making offspring versions of previous generations. The algorithm that was created evaluates the generated maps based on building placement, building alignment, amount of buildings and whether there is a Pokémon Center and Mart, two buildings that are essential to a Pokémon city. After the evaluation the algorithm then sorts the population based on their fitness value and removes the bottom half. It then randomly pairs together the remaining maps and combines them with each other to generate new maps. This combination is done by splitting the two maps along a random horizontal line and removing all buildings along this line that would cause a conflict when the new maps are generated (See figure 1 & 2). When the new maps are created, new buildings are added in order to replace the buildings that were removed in the process.

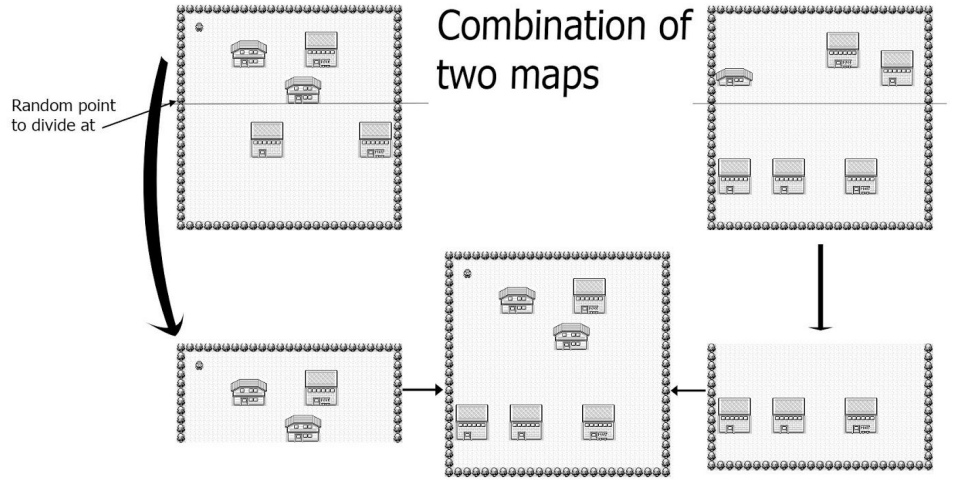


Figure 1. Example of combination of maps

Figure 2. Example of removal of conflicting building

In order to minimize the risk of having the genetic algorithm getting stuck on local maximas, and thereby not being able to further climb the search space, it was necessary to give the generated maps some form of mutation.

Three different mutations was decided upon, which were to either move, remove or add another building to the map. Moving a building would result in a random building to be moved one step in a random direction, removing would take one random building off the map and adding a building would add any type of building available onto the map. Adding a building would only happen if there was enough space available on the map.

When reviewing the result of the generated maps, it became apparent that those which were given a high fitness value sometimes would show to be identical to each other. This lead to the decision of removing any duplicates from new generations of maps.

In order to finalize the maps, structures apart from buildings had to be generated and placed into the cities. Much like the approach of determining how buildings should be placed, the original cities would be analysed in order to determine in which ways it would be possible or

"legal" for the remaining structures to be placed. When these rules were established they were then used to regulate the fitness value of the generated cities.

4.3 Second prototype - Patterns

For the second prototype, patterns from the original game was used as representation. In order to define what could be considered the most simple form of pattern, also known as a micro pattern, the original cities was once again analysed. Through this inspection it was determined that all structures that occured in the cities consisted of small patterns which in themselves contained four games tiles (See figure 3).

Figure 3. Each micro-pattern consists of a rectangle made up of four of the in-game tiles

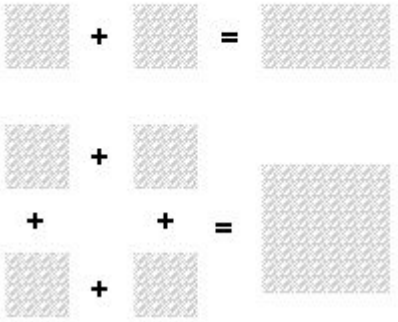


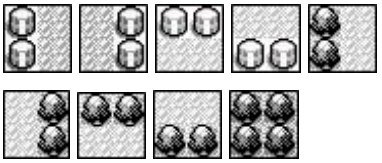
Figure 4. Micro-patterns of the same type can be combined in order to form larger patterns. Here we see two different patterns created with the grass micro-pattern.

As seen in figure four, micro patterns of the same type can be combined in order to create larger versions of each micro pattern. The definition of a pattern in this context is therefore; one or more micro patterns of the same type which are connected together in such a way that they form a rectangle. This means that for every type of micro pattern there are multiple variations of patterns that can be used as structures in a Pokémon city. The one exception where this did not apply was when it came to the buildings and signs, which were pre-defined both when it came to design and sizes. There existed a large variation of buildings in the original Pokémon cities, however, this study only included the three examples shown here due to the limitation of implementation time.

Static Patterns		These four patterns would always be of the same size and without variations.

The other objects which were include was; road, borders, fences, grass, flowers and forests. These objects were always made up of micro-patterns. Fences were limited to expand horizontally and borders only go around the map, all the other could expand in two dimensions as long as the pattern formed was still a rectangle and not too big (max size ~ 4x5 micro-patterns) (See figure 3).

The following table describes all micro-patterns utilized in the prototype.

Road		This micro-pattern would fill out any space not occupied by another pattern.
Border Patterns		These eight micro-patterns created patterns that existed only as a part of the border preventing the player from leaving the city and would extend all the way vertically and horizontally.
Horizontal Patterns		These two micro-patterns created patterns that could extend horizontally within the specified size range. The first of these micro-patterns however, were special in that they would only occur once in its pattern, the rest of the pattern would be aesthetically identical to the second micro-pattern.
Unrestricted Patterns		These three micro-patterns created patterns that were able to extend both horizontally and vertically into a bigger rectangular shape within a specified size range.

In order to define how the generated content was allowed to look, rules were formed based on how the original Pokémon cities were laid out. These rules would define which structures of a city that was allowed to be placed in relation to every other type of structure. This process would eliminate illegal solutions from the search space by making sure that patterns would not be generated in those ways which, in turn, would limit the workload of the genetic algorithm.

This table shows which patterns that are permitted under, over and next to other patterns

Structure	**Permitted under**	**Permitted next to**	**Permitted over**
Building	Grass, Fence, Road, Flowers, Sign, Border	Any	Grass, Fence, Road, Flowers, Sign, Forest
Sign	Any	Any	Any
Fence	Grass, Fence, Road, Flowers, Sign, Building	Any	Grass, Fence, Road, Flowers, Sign, Building, Forest
Road	Any	Any	Any
Flowers	Any	Any	Any
Grass	Any	Any	Any
Forest	Any	Any	Grass, Fence, Road, Flowers, Sign, Forest, Border
Border	Grass, Fence, Road, Flowers, Sign, Building, Forest	Any	Any

With these rules in place a few test runs were run which generated maps that did not meet the level of quality that was expected. This lead to the conclusion that the algorithm would need to have some guidelines in order to be able to more precisely determine what would be perceived as a well designed city. This was done by regulating the fitness value of the generated maps based on what amount of certain structures there were, and weather or not these structures were placed in a way that would be perceived as "good". A list of all the parameters that was used can be found below.

- Average amount of buildings
- Average amount of road
- Average amount of signs
- Average amount of fences
- Average amount of fences under buildings
- Good if the road is connected
- Good if one sign is placed next to a building
- Good if fences are next to buildings
- Bad if signs are next to each other
- Bad if fence is placed under fence

When it came to the point where the pattern representation was finalized it was time to let the genetic algorithm generate new maps with it. This was done in a similar fashion to that of the building representation since the goal was to have an equal playing field when evaluating the final results. One difference from the building representation was that the mutations were changed so that any pattern could be used instead of just the buildings, however, the same mutations was still used, which in this case became moving, removing or adding a pattern.

The main difference between the two prototypes however, was in combining two maps in order to make new ones. This was done by taking the list of patterns from each map and choosing a dividing point at which the lists would be split, the removed parts would then be added to the remaining list of the other map and vice versa in order to create two new maps. Any patterns that conflicted with another pattern in the new maps would then be moved somewhere else.

5 DATA ANALYSIS

5.1 Analysis of efficiency

For our building representation the fitness value of the generated maps did not get better with more than around ten generations. The time needed for those generations to finish were in turn very short, never exceeding even a few seconds.

When evaluating the efficiency of the pattern representation we found that the fitness value of the generated maps were not getting better for a generation count past 200 generations. However, these generations were in themselves taking longer to complete due to their property of having more types of patterns to consider. All together, this lead to a total of around 40 seconds for a finished generated map using this type of representation.

5.2 Analysis of complexity

For the first prototype there was only one type of structure involved in the evaluation, the buildings, while the pattern prototype had several kinds of structures to keep in mind. For this reason the evaluation of the maps generated by the pattern prototype took much longer to get working than for the first prototype. Since the genetic algorithm of the first prototype did not generate any other structures apart from buildings, the remaining structures had to be generated separately. The implementation of this generative function also took a significant amount of time to develop. The differences between generating the surrounding structures separately and having all structures being generated at once were not only limited to the time of implementation, but also had a significant effect on the variation of the resulting maps. These variations were then evaluated by the test subjects in order to determine which method they thought produced the highest quality maps.

5.3 Analysis of quality

For this project we had 21 individual testing sessions and each tester gave feedback both on the maps that they thought were good as well as the ones that they thought were bad. The testers found the maps they gave higher grades to be better laid out, easier to navigate, more aesthetically pleasing, less empty than other maps or, in one case, felt like the buildings were more randomly placed. There were also some positive comments on the placement of signs and the even distribution of structures.

The maps that were given lower grades were given motivations such as being messy, too big considering the amount of buildings, badly structured, full of empty space, plain or having strange roads, lacking shortcuts or roads with no gaps between tiles. Further comments were made on badly placed structures as well as some maps feeling like some structures were randomly placed and feeling like they had less buildings than the others even though they had the same amount as others, this due to all of the buildings being placed in a small area of the city.

Statistics

The results that were collected from the user tests showed that the maps which were generated with the building representation were prefered over the ones that were generated using the pattern representation. On average, the test subjects gave maps from the pattern group a rating of 3.08 out of five, whereas the other group got an average rating of 3.97. When the test subjects were asked if there were any particular reasons they had for giving certain maps a lower or higher rating the answers were a little spread out, although most of them had a common theme. For example, when asked why a specific map was considered below average, the answers were often centered around the opinion that it had too much empty space, or that it didn't feel structured due to a lack of ways to get around in the city. On the other hand, the maps that were given a high rating were perceived as carefully laid out, making them look and feel like they belonged in the Pokémon universe.

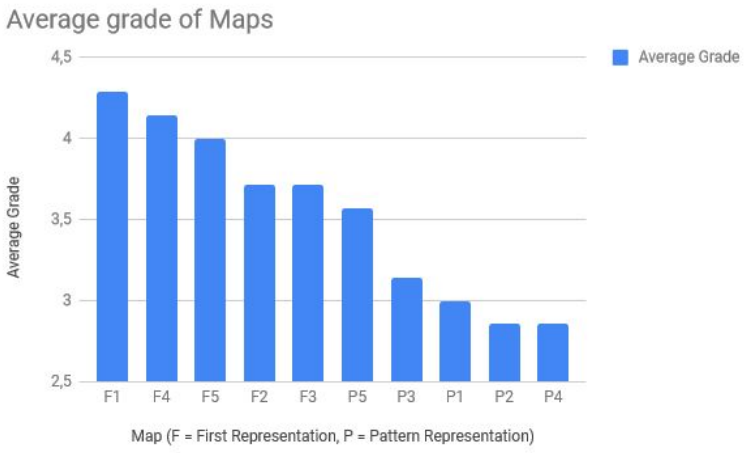


Figure 5. Average grades for all 10 maps

This diagram shows the distribution of the average ratings that were given to each map in both representation groups (See figure 5). Worth noting here is that every map from the pattern group has gotten a lower score than the other groups maps, although one pattern map almost reaches up to the lowest grade of the other group with a score above 3.5.

Comments

Here we present an excerpt of the testers feedback:

Those maps that you rated with a three or lower, what made them worse than the other maps?

I gave city number eight a three because it was so centered towards the right. City number two felt a little messy and city number four felt too big in relation to the amount of houses.

In one of the cities there was a lot of empty space in one particular area, and the road that went downwards could have continued down in order to connect to the other roads. It would have been better if there were more flowers placed out along the way as well.

i found those cities to be very plain and unstructured. Some of the towns had paths that had no gaps between tiles so you'd get stuck and had to go back the same way you came.

Some signs were placed at strange locations, for example between fences, also there were no shortcuts sometimes.

Too much empty space where there is nothing to do or it looks weird.

Some sets of tiles seemed somewhat randomly placed, but not to a degree where I would rate them as "bad" per se.

The buildings were either all on one side of the map, or one type of the buildings were next/close to each other in those cities. When the buildings were concentrated to one point, it made those cities more static. Even if all the cities had the same amount of buildings, you got the impression that these maps had less content.

Those maps that you rated with a four or higher, what made them stand out more than the other maps?

They were nicer and more logically laid out.

The buildings were more packed and while there were still large surfaces of only grass I thought the other parts of the city made up for that. And if you followed the roads there were more flowers along the roadside.

Those cities felt like a actual Pokémon town. Houses and stores where nicely placed. There weren't all that many empty spaces ether which was good

Easy to navigate, signs were pretty ok or well placed. You had a house that had a nice back garden as well. Maps didnt feel empty and good variation

Looks good and have evenly distributed areas to do stuff

The cities with a higher grade seemed well planned out without any strange tile configurations and were easy to walk around in

The buildings are more spread out and are placed more "randomly" in those maps, which makes the cities more dynamic. Contrary to the maps I rated with a 2, these cities seemed to have more content and were more fun to explore.

6. DISCUSSION

6.1 Discussing the results

If we look at the results from our user tests we can see that the pattern representations best result reaches up to about the lowest scores from the building representation. We can also see that the reasoning as of why the maps from the pattern representation (such as in figure 6 and 7) were considered worse than the ones from the building representation mainly was because they felt unstructured and either had lots of empty space or all of its buildings in one area. On the other hand the user tests showed that the maps from the building representation (such as in figure 8 and 9) were considered to have been much better structured with a more logical layout.

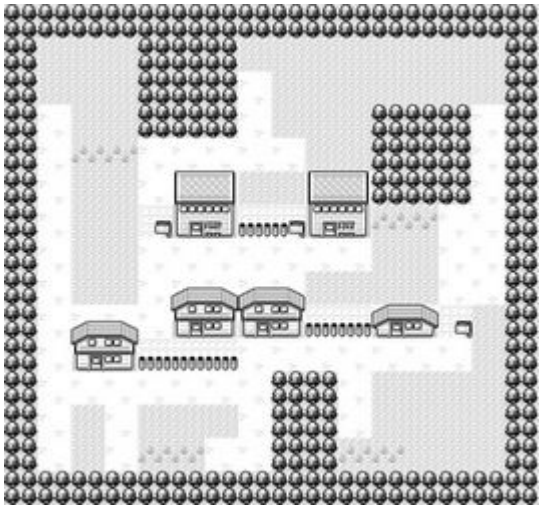


Figure 6. Pattern Representation (best graded)

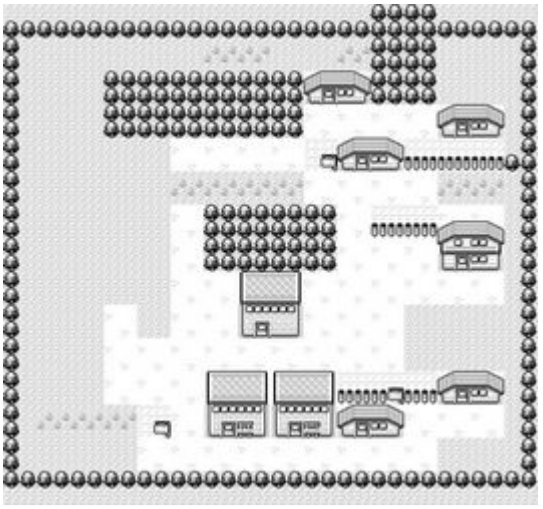


Figure 7. Pattern Representation (worst graded)

The main difference between the representations that were used was the way they handled the generation of the surrounding elements. While the pattern representation generated everything needed in a city using its genetic algorithm, the building representation used a generative algorithm in order to give the city its environment. The way that this generative

method was built made it so that every type of structure had its own priority, meaning that the order in which the structures were built always would be the same. In this case, the order in which the surrounding structures were put into each map were; roads, fences, forests, signs, flowers and grass

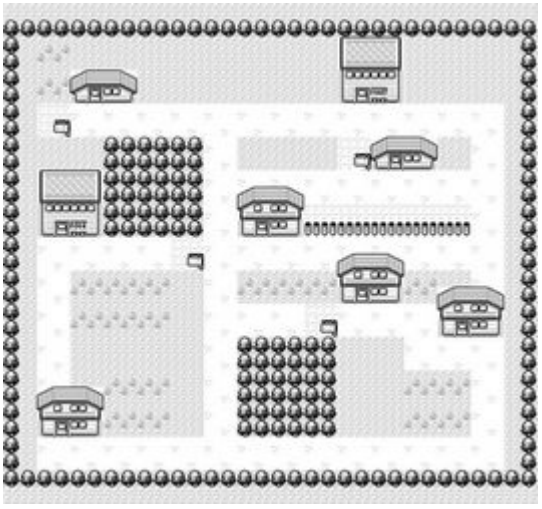


Figure 8. Building Representation (best graded)

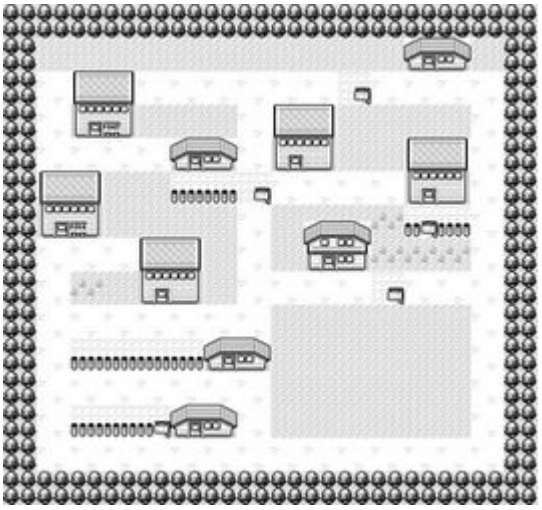


Figure 9. Building Representation (worst graded)

Since the pattern representation made its roads out of the space that did not have any other pattern occupying it, and the generative algorithm made its roads first, this meant that the priority of the roads were completely opposite for each prototype. One thing that was found during the generation of the cities was that the pattern representation would, for each execution, end up with cities where the roads would be almost identical, which most likely was a result of the way it handled roads. Since the evaluation function for the pattern representation was told to prioritise roads that were connected to each other, that possibly made it so the genetic algorithm would only save maps from its population which could be combined with each other without breaking the road network. If for example the priority of connected roads would have been lower in the early stage of the generation process, then the probability of saving maps with other qualities would have been much higher, resulting in a possibility of higher quality maps in the end.

Another possible solution to this problem could have been to not implement the crossover method when generating a new population of maps for the pattern representation and instead only focus on the mutations. This could eliminate problems which would occur when trying to combine maps with each other. Another benefit of this approach would be that it would be possible to generate the roads in an earlier stage and at the same time not have the same risk of losing the road connections.

6.2 Discussing the method

The design science methodology turned out to be well aligned with the goals of this paper. Since it was necessary to develop two separate artifacts it was beneficial to be able to do this iteratively in order to test out how well our solutions worked during the development process.

6.3 Discussing the data collection

In hindsight we established that it would have been beneficial, for the sake of drawing conclusions, if the user testers had been given more specific properties to rate in addition to give each map a general rating. If this had been the case we would have gotten more specific data which would have made it easier to pin down shortcomings for each individual map. This also would have let us narrow down the areas of concern down to specific ones instead of putting all the responsibility on the testers to come up with reasons as to why they thought certain maps were good or bad.

One mistake in this study which was overlooked was that we ourselves made subjective choices when the maps were selected from each representation group. By doing so our own personal judgements had an impact on the final results. If we instead would have generated the exact amount of maps that was going to be used and had the user testers evaluate those maps, then that would have reflected a much more objective opinion, and depending on how those maps would have looked the results might have been shifted around to a certain degree.

6.4 Future work

The earlier study conducted by Luigi Cardamone et.al (2011) concerning representations used in search based PCG was directed towards a 3D environment in the FPS game Cube 2. Their study was used as the basis on which we decided to build upon by extending this research to include 2D environments. It was also a deciding factor to base our research on Pokemon games since search based PCG had been used in another previous study done by Antonios Liapis (2018). However, since this research only focused on determining the relationship between a Pokémon type and its appearance our goal was to extend the research within the field of PCG in Pokémon games to include parts of the gameworld as well.

With this study we have come closer to determine which level of complexity that is needed when it comes to representing a tile based city in a two dimensional gameworld. However, there are multiple ways to further expand the research done in this paper, for example by Including other types of representations, evaluating them on other conditions or by altering the way that the genetic algorithm produces its generations. One example that would be interesting to try out would be to have a genetic algorithm which starts with an empty city and then fills it up with one kind of structure at the time. This could for example be done by running a certain amount of generations for each type of structure until the fitness value would reach an acceptable level for each one. It is of course also a possibility to extend this research to include 3D content, examples of this can be seen in Luigi Cardamone et.al (2011).

It would then be valid to ask the question if using a specific type of content representation would be more beneficial for either 2D or 3D game content.

7. CONCLUSION

This study takes two different representations used in search based PCG and puts them against each other in order to determine which one that performs best in terms of quality, efficiency and complexity. We built two artifacts that each used one of our chosen representations in order to generate cities for the first Pokémon games. In order to determine the quality of the generated maps we conducted user tests in which experienced Pokémon players gave the generated maps individual grades. The evaluation of efficiency was on the other hand based on how long each artifact would take in order to generate the final cities. The test sessions were arranged in such a way that one person at a time would walk through a set of pre-generated cities consisting of five cities generated with each representation in a random order. The test subject was then asked to grade the cities individually, and at the end they were asked to write down their reasoning as to why they gave those grades to the cities.

If we were to only look at the results from our user test, it would be easy to dismiss the pattern representation completely without considering its further potential. Although, during the phase in which we generated the final cities for our user tests we noticed that the cities generated with our building representation were less varied than those generated with our pattern representation. While this was to be expected to a certain degree, it was obvious that in the long run you might want to be able to generate certain types of cities that you would not be able to if you were using our building representation. In our case however, we did not succeed in optimizing the evaluation function for our pattern representation to the level of perfection which would have been needed to mach the consistency of our building representation. Doing further research where these representations are brought up to a more equal ground would make drawing definitive conclusions much more vindicated.

8. REFERENCES

Noor Shaker, Julian Togelius, and Mark J. Nelson (2016). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research.* Springer. ISBN 978-3-319-42714-0.

Steve Dahlskog, and Julian Togelius (2012). *Patterns and procedural content generation: revisiting Mario in world 1 level 1*

Steve Dahlskog, and Julian Togelius (2014). *Procedural Content Generation Using Patterns as Objectives*

Tracy Fullerton (2014). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games, Third Edition*

Mark Hendrikx, Sebastiaan Meijer, Joeri van der Velden, and Alexandru Iosup (2013). *Procedural content generation for games: A survey*

Georgios N. Yannakakis, and Julian Togelius (2011). *Experience-Driven Procedural Content Generation*

Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, Cameron Browne (2011). *Search-Based Procedural Content Generation: A Taxonomy and Survey*

Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, Samir Chatterjee (2008). *A Design Science Research Methodology for Information Systems Research*

Luigi Cardamone1, Georgios N. Yannakakis, Julian Togelius, and Pier Luca Lanzi (2011). *Evolving Interesting Maps for a First Person Shooter*

Antonios Liapis (2018). *Recomposing the Pokémon Color Palette*

José Antonio Leal de Farias. 2009. MonoGame. [ONLINE] Available at: http://www.monogame.net/. [Accessed 1 September 2018].

Jesper Juul. (2012). *A casual revolution: Reinventing Video Games and Their Players*