

1 Samuel Wright (sw734)  
2 COMP6590 Computational Creativity Project Report  
3 School of Computing  
4 University of Kent

5 **Abstract**

6 Welcome to my COMP6590 computational creativity report on my image gen-  
7 eration system, in this report you will find all relevant information on and about my  
8 project, it will include an introduction into my system and project scope, the con-  
9 text and background for my project including theories and related work, method-  
10 ologies and project roadmap showing the stages and design of my project over time,  
11 results showing some outputs, evaluation which explains and abstracts the various  
12 parts of my evaluation system and finally a conclusion explaining my thoughts and  
13 feelings on my project and what actions I would take in the future to further develop  
14 my idea.

15 **1 Introduction**

16 For this computational creativity project, I chose to create a system to generate images,  
17 specifically to generate an image based on a user's sentence description of a scenic view.  
18 This being a place you would see on a walk, either looking out at it or just looking up  
19 into the sky. Basically, whatever the user interprets their scenic view as.

20 To summarize, my system incorporates multiple procedural generation techniques  
21 such as Perlin Noise and an L System to generate the various parts which ultimately  
22 create the image. I implemented natural language word processing in unison with a word  
23 colour association lexicon (which I modified) I found while researching . To finish off I  
24 implemented an evaluation system which allows the user to change parts of the image  
25 to better fit their initial expectations or to alter with their new ideas on what the image  
26 should look like. I will be going more in depth into the full development process within  
27 this paper.

28 **2 Background**

29 **2.1 Natural Language Processing**

30 Before I undertook in this project, I already knew that I was going to use Natural Lan-  
31 guage Processing in some form but was unsure how to properly use it in a way that best  
32 benefits me and my objectives. After reading this article from Oracle I had a much better

understanding on the most common techniques, these being Tokenization, Bag-Of-Words-Model, Stop Word Removal, Stemmatizing and Lemmatization as well as Part-Of-Speech Tagging and Syntactic Parsing [1]. Within my system, after reading through this article thoroughly I decided to include tokenization, stemming and part of speech tagging. I chose these 3 processing techniques as after using them, I concluded that this trio work particularly well together especially for my use case.

When looking into how to best include natural language processing into my project I came to the understanding that python was the best language to use, this article on python and word processing sums it up nicely, “Python, being a versatile and powerful programming language, has emerged as a popular choice for NLP tasks due to its rich ecosystem of libraries and frameworks” [2]. I was also semi familiar with python and word processing at this point, so this gave me a great excuse to get more comfortable with this language and technology.

## 2.2 NLTK vs SpaCy

Now I had a choice to make, what library to include for natural processing? From my background reading it came down to 2 options in the end, these being NLTK and SpaCy. They both have their advantages and disadvantages but after a lot of consideration I decided to go with NLTK. “If you’re working on a small-scale project or need more flexibility, NLTK might be the better choice. If you’re working with large amounts of text and need to process it quickly, or require advanced NLP features, spaCy might be the better choice” [3]. NLTK seemed to be the better choice for my use case as the slower processing and slightly less efficient tokenization don’t hinder the ease of use for my smaller scale project.

## 2.3 Word Colour Association Lexicon

Getting words (or tokens) to be linked to colours in some way was always going to be a struggle, making something like this myself would take way over the time given for this project, so I decided to research into some solutions. Then I came upon the crowd sourced word colour associations lexicon which would work perfectly in my project. One of my main reasons I wanted to find something like this is because I understand the importance of colours within a scene which is summed up nicely in this quote from the paper linked with this lexicon, “using the right colours can not only improve semantic coherence, but also inspire the desired emotional response” [4]. Getting the colours right without the user explicitly stating them was a priority for me and after background reading seemed viable.

## 2.4 Procedural Generation Techniques

Incorporating techniques to generate various parts of the image was a must but finding which ones to use and how to use them effectively became a bit of a setback. With many algorithms to choose from and many resources online to explain parts, I grasped a better understanding. The main learning point was this, “Is there a difference between random generation and procedural generation? These two terms are often used interchangeably. From my understanding, random generation is a part of procedural generation, but procedural generation is not always random” [5]. I intended to use some sort of randomness in my procedural generation, but I needed to remember not to go overboard with it as it can take away from desirability and accuracy of procedurally generating the same output twice which is something interesting to consider.

When abstracting at looking at the smaller parts of procedural generation, one of the important parts that come to mind is tree generation, how do I generate trees that are inheritably the same but have slight variation? Well, I came to the conclusion on using an L – system to achieve this, I won’t go into much detail now as I intend on doing that later but there was an interesting point I came upon while researching. That being “The L-system is a particular type of fractal that can be used to model trees and plants. It was first conceived as a mathematical model to describe the growth and interaction of cells within plant structures” [6]. I intend to use this algorithm in the context of tree / plant generation. Similar can be applied when using something such as Perlin Noise. When doing background reading, I came upon the significance between Perlin Noise and White Noise, “Its mathematical foundation lies in generating gradients across a grid and interpolating them, which distinguishes it from ”white noise,” where each point is entirely random and lacks continuity. [7]” These are both functions I considered to include because of this.

## 2.5 Existing Work With Procedural Generation

When looking through existing examples of procedural generation there was one that stood out to me the most, Minecraft. And to my surprise it held a lot of similarities with my way of thinking and way of applying algorithms for content generation. Although it is not necessarily using it for image generation there are still some good pointers to take from it. For example, “the game makes use of gradient noise algorithms, like Perlin noise. This makes sure blocks and chunks fit with its neighbours and gives the world both continuity and randomness” [8].

## 3 Methodology And Design

To design this system and to achieve a successful output, I had to think smaller as there are so many moving parts in order to get this into a functional state, so I began where I always tend to begin. Abstraction. I knew if I was going to achieve something I felt proud about that I would need to look at every part individually (for the time being), the way I did this was secluding each part of the system's logic in individual files and bringing the system as a whole into one main file. This was a foundational principle for me, especially it being the first time I have attempted to create my own complex system.

### 3.1 System Design

To begin with, I implemented the logic of dissecting the user's input into tokens using NLTK. Then I removed any punctuation, turned all tokens to lowercase, tagged the tokens and ran it through my lemmatization technique function. This entails checking the tag associated to the token (for example noun) and lemmatizing accordingly. Following this hierarchy of processing techniques allowed me to follow the correct sequence as well as following the correct procedures such as getting rid of punctuation and changing to lowercase.

Next, the tokens are passed onto the next process of my system with this being associating a colour(s) with each token. To achieve this for starters I knew that there were only 12 colours included within the Lexicon, so I associated a hex and RGB value to each colour within the lexicon in their own dictionary data type. From here the tokens are passed into the `colour_association` function which scans through the lexicon, checks whether the token is found within the lexicon, get the associated colour name if it is included, change that colour name into its RGB value and then attach that RGB value to the associated token.

Once the colours have been associated, all information is passed to the biomes file which is the bread and butter of the system so to speak. Firstly, there is a list of biomes that can be generated by their corresponding functions as well as a synonyms dictionary which maps like words to the words that are mapped in the list to biomes (I made sure every synonym is present in the lexicon beforehand). Once the information is passed into this file the first function called attached the colour and token to the corresponding biome (if matches).

The image needs to be generated as something, so by using Object Oriented Programming (OOP), I made a Canvas class which can be used to make canvas objects.

And for the final part of the system is the evaluation, I will not go to into depth as this part follows soon but there is evaluation in place so the user can decide if it meets

135 their expectations and if it doesn't then they have the option to alter the image to best  
136 fit their expectations.

## 137 3.2 Limitations

138 When adding more and more parts to the system, it began to get complicated. For  
139 instance, as I begun adding more and more biomes it got increasingly difficult to find a  
140 way to generate the biomes without conflicting one another.

141 Using the PIL library (for canvas) might have been both a gift and a curse. It is a  
142 fantastic way to get all parts of the image generated onto an object that makes sense for  
143 my use case but there is very little for what can be done to modify parts of the image (via  
144 the evaluation). I won't go too much into this now as I have a solution to be discussed  
145 at a later point.

## 146 4 Results

147 Here are some example outputs given from my project; these can be saved into the project  
148 directory upon user's approval.

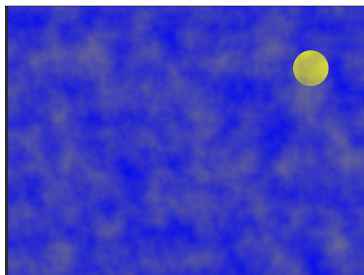
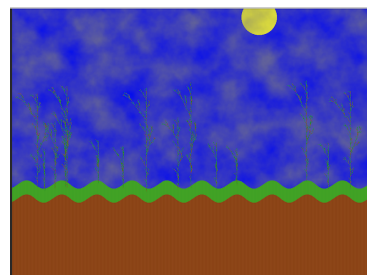


Figure 1: **Pure sky:**



**Terrain Generation:**

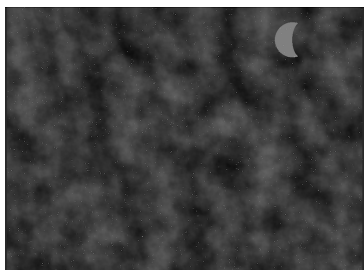
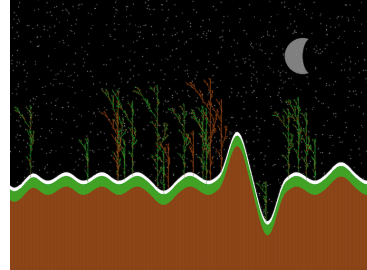


Figure 2: **Pure Night Sky:**



**Terrain Generation:**

149 Image 1 was generated using this prompt: Sunny skies with clouds all over and  
150 image 2 using Sunny skies with clouds all over with mountains with trees on  
151 them. Image 3 and 4 are generated in the same way via shared tokens in their prompts,

152 this goes to show that no matter if the tokens are the same the output will always have  
153 some sort of variation.

## 154 **5 evaluation**

155 On a base level, when looking at the system I created it indeed meets the criteria of  
156 generating an image based upon a user's input / description. But this is not good  
157 enough. To delve deeper and find out if this system truly serves its purpose you need  
158 to use some sort of basis for evaluation. I have decided to use Simon Colton's creative  
159 tripod for evaluating the computational creativity of my system. This will be done by  
160 measuring if it has met these 3 criteria's, skill, appreciation and imagination.

161 In the end when evaluating via Colton's Creative Tripod, my system does not fully  
162 meet the requirements for the title of creative. The short fallings being imagination,  
163 one way to tackle this would be to add random effects such as a solar eclipse, shooting  
164 stars or rainbows (each with a small percentage chance of appearing). With this addition  
165 implemented I believe then it would be deemed as 'creative' and add the much-needed  
166 novelty to the system.

### 167 **5.1 Skill**

168 Skill is defined as the technical competence of the system, basically how well the system  
169 creates images. So, to an extent skill is shown within the scope of my system. When  
170 generating images of purely the sky parts (with no terrain generation) the system shows  
171 clear skill, especially in the generation of the clouds and also in the layering of parts such  
172 as the moon, clouds and stars. Furthermore, the trees generated show some promising  
173 artefacts of skill, but the terrain generation left for more to be desired.

### 174 **5.2 Appreciation**

175 Appreciation is how well the system can be understood, both by others and by itself.  
176 When evaluating this system, I decided to implement an appreciation function into the  
177 system. This allows the user to give feedback for the system to respond and change to  
178 accommodate for the users' actions (at a limited level). Although my system does not  
179 have any fitness function allowing it to self-evaluate, I believe my appreciation function  
180 to be a promising attempt and sufficient.

### 181 **5.3 Imagination**

182 After a lot of consideration, I deem my system to not be very imaginative or novel. In  
183 contrast, to an extent there is some novelty to the system as no two images will be exactly

the same even when given the same prompt, but this can be argued of walking the fine line between procedural and random generation. If you have read my proposal for this project on the other hand, I believe that my system coincides with my inspiration of generating scenic images I would like to see on walks.

## 6 Conclusion

To conclude my thoughts and feelings from this system and the project as a whole, I believe that being self-critical and being able to see where my system falls short will help me in the future (if I were to continue development). I feel I have achieved a valiant attempt at building a ‘creative’ system with the time given and with the suggestions that will follow I believe I could build upon the foundries I have already implemented.

### 6.1 Learning Outcomes

From this project, I have learned about the various processing techniques adopted by natural language libraries and the advantages and disadvantages of them.

I have learned about how to implement as well as tinker with procedural generation algorithms to better suit my needs.

I have learnt how to use existing tools (such as the lexicon) and how to better adapt them to suit my use case.

I have learnt and understood how to critically evaluate my system, both by using Colton’s Tripod System as well as my user evaluation function built into my system.

### 6.2 Improvements

To end on, I have compiled a list of improvements which I believe would make my system excel in every way and define it as a truly creative system. If the opportunity arose these are the improvements that I would make.

I believe at times, I may of blurred the lines between procedural and random generation, I was definitely heading in the right direction by adding randomness to the generation but I made no way of generating the same image twice. I could of taken a note out of Minecraft’s book and used bit mapping to be able to generate the same image again (as I referenced in my proposal).

The lexicon only having 12 colours associated within it was a limiting factor, if I had an infinite amount of time I could create my own lexicon using the full range of colours (or see if there is scaffolding of one online).

I would like to go back and improve how the user evaluation function works, specifically I would like to make it easier to alter the existing parts of the canvas. I spent a lot of time

trying to make this work but with no success, I believe the best option to be making this system a hybrid of python for all the pre and text processing steps and then generating the canvas in something like Processing (JavaScript IDE). I did consider this, but I thought I would challenge myself and be ‘creative’ by thinking outside the box.

Adding some sort of UI would have been ideal (but was not a necessity given the time scale), it would make the system much more user friendly as I understand that without computer knowledge it would be hard to run this system in it’s current state.

So to conclude this paper, although it may be limited by the ‘creative’ aspect of the evaluation. I still very much believe this to be a great start of system that has the potential of becoming far greater, I hope you enjoyed reading this as much as I had writing and programming it.

## 7 References

[1] - Caroline Eppright (2021), What Is Natural Language Processing (NLP)?, [https://www.oracle.com/uk/artificial-intelligence/what-is-natural-language-processing/#:~:text=Natural%20language%20processing%20\(NLP\)%20is,natural%20language%20text%20or%20voice](https://www.oracle.com/uk/artificial-intelligence/what-is-natural-language-processing/#:~:text=Natural%20language%20processing%20(NLP)%20is,natural%20language%20text%20or%20voice)

[2] - Trantorindia (2025), Natural Language Processing with Python: A Beginner’s Guide with Example Code and Output, <https://www.trantorinc.com/blog/natural-language-processing-with-python>

[3] - Prabhu Srivastava (2023), SpaCy vs. NLTK: A Comprehensive Comparison of Two Popular NLP Libraries in Python”, <https://medium.com/@prabhuss73/spacy-vs-nltk-a-comprehensive-comparison-of-two-popular-nlp-libraries-in-python-b66dc477a689>

[4] - Saif M. Mohammad, Even the Abstract have Colour: Consensus in Word–Colour Associations, Institute for Information Technology National Research Council Canada., <https://aclanthology.org/P11-2064.pdf>

[5] - Kenny (2021), Procedural Generation: An Overview, <https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41>

[6] - Samal Et Al (1994), ng Plants Using Stochastic L-Systems, University of Nebraska - Lincoln, <https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1036&context=cseconfwork>

[7] - Garage Farm.NET, Perlin Noise: Implementation, Procedural Generation, and Simplex Noise, <https://garagefarm.net/blog/perlin-noise-implementation-procedural-generation-and-simplex-noise>

[8] - Minecraft Wiki, World generation, [https://minecraft.wiki/w/World\\_generation](https://minecraft.wiki/w/World_generation)