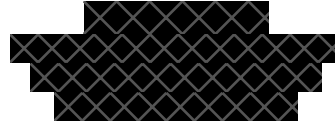
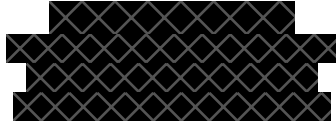


Pikazard: I Choose You!



Abstract

Procedural Content Generation (PCG) is used for creating content for games. Useful content is novel, follows existing game aesthetics, and is representative of game mechanics. We develop a creative system that procedurally generates Pokemon from existing Pokemon assets. Novelty is handled indirectly by our system while its main design is oriented around creating Pokemon that reflect a given input of traits and uses heuristics to follow Pokemon aesthetics. A survey of how well the generated Pokemon represent the given input is evaluated. The system shows potential with a lot of room for improvement.

Introduction

Procedural content generation (PCG) has been leveraged by game studios for producing game content at the industry level. PCG has been applied in several ways at varying scales. As summarized by (Wikipedia contributors 2022), PCG has been used for music, weapons, creatures, and most common on the list, 2D and 3D world terrains. This list consists of only games that leverage PCG as a dynamic aspect of the game, meaning content generation is done at “runtime”. Games where PCG is used only in development, to aid asset creation, are not included, making the list a lower bound on the presence of PCG influence on games. For PCG to be helpful the content generation can be guided to give meaningful results. Meaningful results ultimately hinge on the target experience the game is aiming to provide the player. To be more concrete, generated content needs to fit within the asset constraints of the game, such as aesthetics and themes, game difficulty balancing, and functionality requirements yet provide novelty within those constraints.

Content created that does not follow the existing style or is too “extreme” in variance will appear out of place, degrading the experience, but generated content that replicates (or varies very little from) existing assets does not provide value for a unique player experience nor offer a developer inspiration. In other words, the desired target is the peak of the wundt curve (Berlyne 1973), a ‘not too much not too little’ amount of novelty. In the case of our system, more specifically, a creature that differs from existing Pokemon yet would still be recognized as a Pokemon is the aspired result for novelty.

PCG additionally needs to account for “sub text” of the content it is creating. Well designed assets effectively communicate value, utility, functionality, and even game state to players. A generated environment may need to incorporate a visual for platforms that damage the player in order to differentiate from platforms that are neutral. Music that is procedurally generated can take into account and communicate the game state; playing pleasant music based on positive progress or alter to a dissonant tone in the case the players vital resources are nearing empty. A generated gun that has a longer effective range and accuracy could visually be given a longer barrel whereas a more mobile automatic gun would have a smaller stock and a bigger magazine. An asset can be considered better than another if it more clearly communicates to players relevant game information.

Pokemon Generation

Our work is to build a creative system which procedurally generates novel Pokemon that follow implicit rules relating a Pokemon’s traits, strengths, and weaknesses. The weapon example, in the previous paragraph, is most similar to the role which Pokemon have in their game mechanic. Pokemon have “stats”. From Bulbapedia, “A stat is any of certain numerical values pertaining to each Pokemon. The Pokemon stats are used in battles. This is short for statistic; in some cases, it has also been named ability, rating, effect, or parameter (Bulbapedia contributors 2022a)”. Each Pokemon has 6 permanent stats which we will continue to refer to as stats in this paper. Every Pokemon has one or more Types. A Type is a set of properties applied to a Pokemon and its moves, affecting their performance in battle, “the Types are based on the concept of classical elements in popular culture (Bulbapedia contributors 2022b)”. See Figure 1 for examples of Pokemon with their stats and Figure 2 for a list of all Pokemon Types.

For PCG to create novel and ability representative Pokemon we leverage Pokemon image assets, Type, stats, and other data from existing Pokemon to form new ones. As such we stay within the domain of existing Pokemon and the system navigates within that context; presumably, able to follow the implicit trends fostered by the original Pokemon creators.




#	Pokémon	HP	Attack	Defense	Sp. Attack	Sp. Defense	Speed	Total	Average
001	 Bulbasaur	45	49	49	65	65	45	318	53
002	 Ivysaur	60	62	63	80	80	60	405	67.5
003	 Venusaur	80	82	83	100	100	80	525	87.5

Figure 1: Three example Pokemon with there respective stats (Bulbapedia contributors 2022c)



Figure 2: List of all Pokemon Types. (Bulbapedia contributors 2022b)

Approach

To accomplish our goal of creating a system that is capable of procedurally generating Pokemon, we have designed a ‘nearest neighbor’ algorithm with two levels of sampling complication. Our system is able to find the relevant sampling pool using the user-supplied Pokemon Type and select body parts from the most similar Pokemon, using stats as the metric for similarity, to the desired Pokemon. This relevant sampling pool will be expanded on in a later section, but serves as an additional layer of relationships between Pokemon that can be used for selecting nearest neighbors.

Pokemon Dataset

In order for our generated Pokemon to have a degree of variety, we needed a large dataset of Pokemon parts for our system to choose from. Doing this ensures that the same body parts don’t get selected too often, which risks visually stagnating the generated Pokemon.

There are currently eight generations of Pokemon and nearly 900 unique Pokemon in total. We chose to limit ourselves to only the first three generations, or 386 Pokemon, due to the fact that the sprites for the first three generations were of a low enough resolution that pasting together pieces of different Pokemon could still look somewhat natural. Using more high resolution sprites would have potentially yielded Pokemon that had very obvious ‘seams’ between body parts. This would have detracted from the resulting Pokemon image, as we are aiming for domain competence.

Each of the 386 Pokemon sprites and data were pulled from the PokeAPI Python library. After the full sprites had been collected, we went through each sprite and isolated and saved each of the Pokemon’s legs, arms, heads, tails, and bodies (see Figure 3). With this done, we next needed to crop each body part’s image (except for the Pokemon’s bodies themselves) from their original 64x64 dimensions to the dimensions of the respective body part. In order to do this, we used the Python library CV2 to apply a thresholding operation to the isolated body parts and to find the resulting bounding box, then used the coordinates of that bounding box to extract the cropped body part, isolating it from the rest of the 64x64 image. An example of this operation is shown in Figure 4.

This needed to be done in order for the pasting of body parts onto a Pokemon body to work properly, as each body part’s image dimensions needed to be centered to itself in order to be pasted into the correct location.

The final piece of preparation that needed to be completed before our system could function was to create body templates for the system to paste Pokemon body parts onto. In Pokemon, there are 14 unique body shapes. These include ‘upright’, ‘quadruped’, ‘ball’, ‘tentacles’, and more. The Pokemon’s body shape dictates what body parts that Poke-

mon’s body has attached to it, and it also dictates what body parts it contributes to the overall dataset.

In order for our system to paste the isolated Pokemon body parts (head, legs, arms, etc.) onto the isolated Pokemon body in a cohesive manner, it needs ‘anchor points’, or (x,y) coordinates that tell the system where to put the arms, legs, head, etc. These anchor points must be generated manually for each Pokemon body that we wish to include in our system’s sampling. It was for this reason, among others, that we opted to include only two body shapes for our system to generate: ‘upright’, and ‘quadruped’. Another reason we narrowed it down to these two body shapes was because these two body shapes occur the most commonly of the 14 body shapes in the first three generations of Pokemon. This gave us the most examples to select the Pokemon whose bodies best lend themselves to being isolated and having other Pokemon parts pasted to them. For each of the two body types we used, we selected the two or three Pokemon whose sprites best fit this description. We took their isolated bodies and manually found the coordinates for where the head, arms, legs, and tails should be pasted. Our system uses these coordinates to attach the isolated body parts to the appropriate body.

K-Nearest Neighbors

K-nearest neighbors (KNN) is a method to classify an unlabeled data point in n-dimensional space. By calculating the distance, using a chosen distance metric, of the unlabeled data point with the labeled data, a ranking of data points is made by order of least distance. Then, from the highest k ranking points, the unlabeled point’s class is estimated as the most common class of the k-nearest points. For determining shape of the Pokemon we employ KNN, where the shape is the class we are determining for our unlabeled input data, and we find the distances from the subset of the data that has the same Pokemon Type as given in the input. We mention that currently our system only supports the quadruped and upright body shapes because we still need to label (x,y) coordinates for anchor points of the necessary body attachments. When KNN classifies a shape that our system does not yet support, the chosen shape defaults to upright. For the distance metric we use the Euclidean distance of all the numerical Pokemon battle related dimensions, applied as shown in equation 1.



Figure 3: A Pokemon being separated into its base members

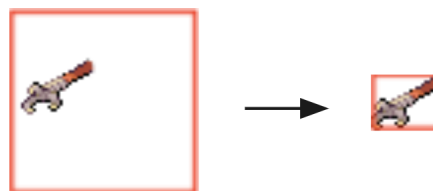


Figure 4: A Pokemon arm asset being reduced from 64x64 dimensions to the threshold dimensions

$$dist(x, t) = \sqrt{\begin{aligned} &(HP_x - HP_t)^2 + (Att_x - Att_t)^2 \\ &\quad + (Def_x - Def_t)^2 \\ &\quad + (Sp.Att_x - Sp.Att_t)^2 \\ &\quad + (Sp.Def_x - Sp.Def_t)^2 \\ &\quad + (Speed_x - Speed_t)^2 \end{aligned}} \quad (1)$$

Where x is the instance from the data set and t is the target input to be classified. After the shape is chosen the system looks through the Pokemon we have templates for in that shape and chooses the one that is closest to the target input using the Euclidean distance in equation 1. Then the next steps are to iterate through each of the body parts required by the selected template.

Egg groups

In the game of Pokemon there exists a game mechanic of breeding. With this mechanic, a player can supply two different (one male, one female) Pokemon which may produce an egg, that will later hatch into the provided female Pokemon but can inherit abilities of the father. For breeding to have a chance of successfully producing an egg, it is required that the breeding Pokemon be of the same egg group (Bulbapedia contributors 2022d). Egg groups are given to Pokemon based on their biological traits, or in other words, physical appearance. For example, the Water 1 egg group



(a) Upright Pokemon Template



(b) Quadruped Pokemon Template

Figure 5: Example templates: The body asset is taken and the appropriate (x,y) coordinates (marked with blue circles) are chosen as anchor points for the necessary body parts of the particular body shape.



Figure 6: Egg groups (Bulbapedia contributors 2022e).

comprises of Pokemon amphibious in nature, the Water 2 egg group Pokemon are piscine like, and Pokemon in Water 3 egg group resemble aquatic invertebrates (Bulbapedia contributors 2022e). Like Type, a Pokemon may belong to 1 or 2 egg groups.

We leverage these egg groups to add diversity in a controlled way. First we iterate through all of the Pokemon of a given Type, keeping a tally of each egg group that appears. This results with a distribution of egg groups as shown by the example for Fire Type in Figure 7. With the relevant distribution, for each of the necessary body part sets (arms, head, etc.), we sample the distribution getting an egg group. Then, for each egg group we get the subset of Pokemon belonging to the particular egg group and find the closest one to the target using the Euclidean distance metric (equation 1), giving us all the parts to assemble our new Pokemon (see Figure 8). This manner of choosing Pokemon parts stays within existing Pokemon biology Type patterns (e.g. No Fire Types with Fish parts), being weighted towards the more common egg group, but still allowing for a mixing of body parts from different egg groups.

Pasting Pokemon

With the body template as a base, body parts are pasted onto the template following an order that gives a natural layering in the event that there is some overlap of the collaged assets. One instance where this has a large influence on the coherence of the Pokemon is with the head, which is always appended to the template last. An additional challenge of placing the body parts is that body parts vary in size and shape. To deal with this, we have generally applied heuristics for placing each of the body parts with respect to the templates' anchor points.

Results and Analysis

We wished to evaluate our system's ability to generate Pokemon that well represent user-provided descriptive stats. To accomplish this evaluation, we conducted a survey that

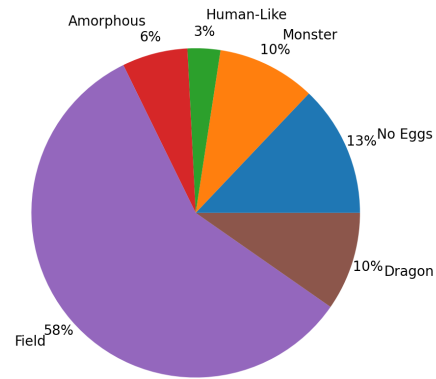


Figure 7: Fire Type Egg Group Distribution. Note that there is no Fairy, Flying, Water 1, Water 2, Water 3, Mineral, Bug, Plant, or Ditto.

showed participants five different Pokemon that were generated by our system and the descriptive stats that our system used to generate them (health, attack, defense, special attack, special defense, speed, and Type). The participants were asked to rate on a scale of 1 to 10 how well they believed that the stats described or represented the generated Pokemon. They were also asked to rate their familiarity with Pokemon creatures on a scale of 1 to 10. This was asked to give some context to individual responses.

The Pokemon images that participants were shown in the survey are given in Figure 9. Three images were shown for each Pokemon. The first is a multi-colored image that shows each piece pasted onto the body without any alterations made. The second is a grayscale version of this image. The grayscale image makes it harder to tell where the seams are between body parts and the underlying body, making for an image that feels less stitched together and more cohesive.

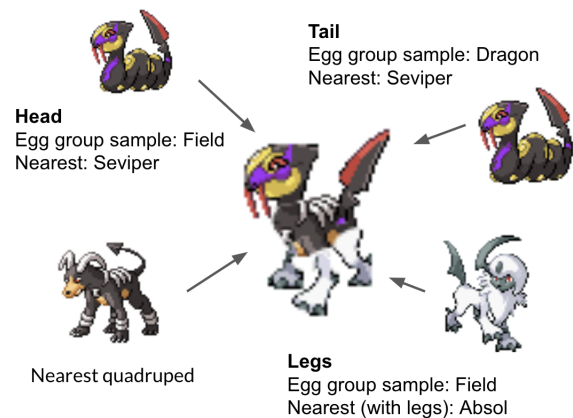


Figure 8: Example showing the nearest Pokemon of each group composing the newly assembled Pokemon.



Figure 9: Pokemon Images used in Survey in ascending order (e.g. top row is Pokemon 1).

The third is a tinted version of this grayscale image. The color of tint applied corresponds to the Type that the generated Pokemon has been designated as by the user’s input. We have pre-selected colors for each Pokemon Type that we feel best represent that Type. For example, a grassy green for Grass Type Pokemon, and a soft blue for Water Type Pokemon. The tinted image is not meant to be especially informative for the participant taking the survey, it is mostly meant to emphasize what Type the Pokemon is.

Our survey was distributed through the use of a Google Form. We used a combination of convenience and snowball sampling to reach as many people in our respective spheres as possible. A total of 27 responses were received. The average familiarity with Pokemon creatures was 5.413, with a median familiarity of 5. The average ratings received for each of the five generated Pokemon the participants were shown is given in Figure 10

These results show that the participants considered Pokemon 3 and Pokemon 4 to be the best of the five Pokemon at representing their accompanying stats, with Pokemon 2 being considered the worst. We believe that one potential rea-

	Average Rating (1-10)	Median Rating (1-10)
Pokemon 1	5.621	6
Pokemon 2	5.517	6
Pokemon 3	6.517	7
Pokemon 4	6.517	7
Pokemon 5	6.275	6

Figure 10: Survey results showing average ratings for each Pokemon in the survey.

son that Pokemon 3 and Pokemon 4 finished first is that they have pieces that fit very well with their bodies, so aesthetically they look more natural, which could be giving their ratings a boost, even without looking at their stats. With consideration to their stats, however, we believe that the reason that Pokemon 3 finished highly was because it had very balanced and middling stats for the most part. It also had no dominant color among its selected body parts that would suggest a Type other than its intended Type of Normal. We believe that Pokemon 4 received high ratings for a similar reason, that its body parts are very suggestive of a Water Type Pokemon, which was the intended Type for that Pokemon.

In addition, the reason that we think Pokemon 2 finished in last place among the ratings was because of its Flying Type. In our dataset, almost all Flying Pokemon had wings of some sort, or otherwise resembled a creature of the air. Our Flying Type generated Pokemon had no such features. Pokemon 2 also had the least cohesion among its body parts when compared to the other Pokemon. Its arms and legs are both hovering away from the body. This lack of cohesion, along with any features that could distinguish it as a Flying Type Pokemon, are the reasons that we believe caused it to finish last in the ratings.

One weakness that we have found in our survey is that it seems that adherence to the intended Type of the generated Pokemon held more weight in the participants’ ratings than the numeric stats. In the future a survey could potentially be conducted that does not reveal the intended Type of the generated Pokemon as a way to prevent this. However, despite this, we consider this survey’s results to be confirmation that our system is capable of generating Pokemon that well represent the descriptive stats that were used to generate them.

Another weakness of our survey is that we hand-selected the five Pokemon that were shown to the participants. They were selected based on how coherent their images were, meaning all of their body parts are connected and look like they fit. In other words, they represent some of the best looking Pokemon that our system is capable of generating. This is a weakness because it means that the survey is only evaluating our system under optimal conditions. If we wanted to evaluate our system holistically we would need to include the generated Pokemon that are not as cohesive. However, we still feel that these survey results are valuable and that reasonable conclusions can be drawn from them.

Common Errors

Our system, while effective at generating Pokemon, is capable of producing poor results. Some of the common flaws

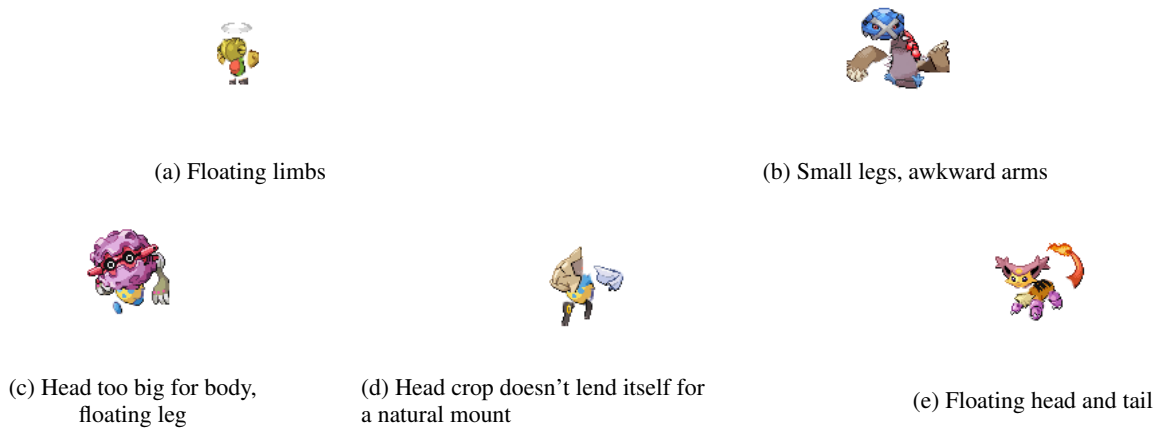


Figure 11: Generated Pokemon with Issues

that appear in the generated Pokemon include body parts that are not attached convincingly to the body, with white space appearing between the piece and body.

Another flaw that appears often is body parts being either being too small or too large for the body to which they are attached.

We also discovered an interesting flaw in the generation of Bug Type Pokemon. When generating a Bug Type Pokemon, there is an extreme lack of variety among the results. This happens because of our egg group sampling method. In our dataset, there exists little to no overlap between Bug Type Pokemon and other Types of Pokemon in the same egg group. This means that when a Bug Type Pokemon is generated, it is choosing from a very small sampling pool of potential Pokemon parts. This is what causes the lack of variety among generated Bug Type Pokemon.

Finally, a common flaw that we noticed occurs in our system when the majority of the user-supplied stats are maximized or minimized, meaning most or all of them are at either extreme. Doing this biases the system towards a select few Pokemon that have stats near these extremes, a direct result of the reduced number of nearby Pokemon due to the extreme stats. We achieved our best results with stats that lied around the middle of the ranges for each stat.

All of these errors are a result of the quality and size of our dataset. Improvements such as the ability to scale body parts up or down to match the size of the body they are being attached to, or an improved method of attaching body parts to a body to eliminate white space would fix some of these issues. A larger dataset would add more data to the smaller sampling pools that we have encountered. With a larger and improved dataset, these errors would be mitigated or eliminated completely.

Contributions and Future Work

It is our belief that, per Jordanous's "key components of creativity" (Jordanous 2011), our contribution to the field of computational creativity includes:

- Variety, divergence, and experimentation

- Originality
- Value
- Creation of Results
- Domain competence

Our system is capable of generating interesting and new combinations of pieces of existing Pokemon using a novel scheme for creating a sampling pool from which the nearest Pokemon to the users' inputs are drawn. While the assets that we use to create our Pokemon are not original, the resulting combinations of them are. Additionally, due to our non-deterministic method of sampling egg groups for Pokemon parts, our generated results have a high degree of variety. We believe that all of these aspects provide value to the field of computational creativity.

Despite all of the features we were able to include in our system, there are many things that we would like to put forth as future work that could be done to improve it. First is a larger and improved dataset. We opted to only include the first three generations of Pokemon due to time constraints and due to the fact that, after the third generation, the Pokemon's sprites began to be higher resolution, which we felt did not lend itself as well to our idea of isolating and pasting together body parts. A larger dataset could be obtained by waiving this second concern, or potentially by programmatically scaling down higher resolution Pokemon sprites to match the resolution of the sprites we used (64x64 pixels), at the cost of some detail. Including a larger dataset would increase the variety of generated Pokemon by increasing the pool from which Pokemon parts are selected. It would also allow us to increase the number of Pokemon body shapes we include in our system, which is the second future addition we would like to see implemented. With the addition of more Pokemon, the total number of Pokemon for each of the 14 body shapes would also increase, meaning finding a Pokemon sprite whose body would lend itself well to having Pokemon parts pasted onto it would be more likely.

Beyond our dataset, we would also like to see additional features added to our system. One such feature is the po-

tential for deviations away from our pre-programmed templates. Currently, our system only allows for the body parts that it has been programmed to seek out, and only permits the number of them that the corresponding Pokemon shape has. For example, a quadruped can only have four legs, and cannot have wings. We think that a system that could adjust these parameters in an organic way would yield interesting results, such as upright Pokemon with multiple heads, or a quadruped with wings. These combinations are currently not possible with our system, and we think that a less rigid formula for generating Pokemon would allow for a higher degree of variety among generated results.

Another area for potential improvement could be to do away with using pieces of existing Pokemon. One aspect of our system that could be considered a weakness is that all of the Pokemon that the system generates are made from pieces of existing Pokemon. This has the advantage of staying in the domain of Pokemon that already exist, but prevents the creation of completely novel Pokemon. The ability for the system to generate Pokemon that conform to the domain of existing Pokemon while generating new ones would be valuable, but difficult to implement, and would require a completely different strategy than what we used.

Aside from improvements to our system, future additions could include features such as name and description generation for the generated Pokemon using natural language processing (NLP). Adding this would increase the system's domain competence, making the Pokemon generated by the system feel more organic when compared to the Pokemon in our dataset.

Conclusion

We have created a system that is capable of procedurally generating novel and cohesive Pokemon using a nearest neighbor algorithm and pieces of existing Pokemon. We believe that our system demonstrates a novel method of accomplishing procedural generation and provides an interesting scheme for finding nearest neighbors to the users' inputs. Further improvements could be made to the underlying dataset that is used for constructing the Pokemon, but our system in its current state represents an interesting and valuable contribution to the field of computational creativity by demonstrating variety in the generation of novel Pokemon and value in these works. Our survey showed that our generated Pokemon, with optimal conditions, can be considered to be competent in their domain.

References

- Berlyne, D. E. 1973. Aesthetics and psychobiology. *Journal of Aesthetics and Art Criticism* 31(4):553–553.
- Bulbapedia contributors. 2022a. Stat — Bulbapedia, the community-driven pokémon encyclopedia. [Online; accessed 18-April-2022].
- Bulbapedia contributors. 2022b. Type — Bulbapedia, the community-driven pokémon encyclopedia. [Online; accessed 18-April-2022].
- Bulbapedia contributors. 2022c. Type — Bulbapedia, the community-driven pokémon encyclopedia. [Online; accessed 18-April-2022].
- Bulbapedia contributors. 2022d. Type — Bulbapedia, the community-driven pokémon encyclopedia. [Online; accessed 18-April-2022].
- Bulbapedia contributors. 2022e. Type — Bulbapedia, the community-driven pokémon encyclopedia. [Online; accessed 18-April-2022].
- Jordanous, A. 2011. Evaluating evaluation: Assessing progress in computational creativity research. *Proceedings of the 2nd International Conference on Computational Creativity, ICCO 2011*.
- Wikipedia contributors. 2022. List of games using procedural generation — Wikipedia, the free encyclopedia. [Online; accessed 16-April-2022].