Recomposing the Pokémon Color Palette

Antonios Liapis

Institute of Digital Games, University of Malta
antonios.liapis@um.edu.mt,
WWW home page: http://antoniosliapis.com/

Abstract. In digital games, the visual representation of game assets such as avatars or game levels can hint at their purpose, in-game use and strengths. In the Pokémon games, this is particularly prevalent with the namesake creatures' type and the colors in their sprites. To win these games, players choose Pokémon of the right type to counter their opponents' strengths; this makes the visual identification of type important. In this paper, computational intelligence methods are used to learn a mapping between a Pokémon's type and its in-game sprite, colors and shape. This mapping can be useful for a designer attempting to create new Pokémon of certain types. In this paper, instead, evolutionary algorithms are used to create new Pokémon sprites by using existing color information but recombining it into a new palette. Results show that evolution can be applied to Pokémon sprites on a local or global scale, to exert different degrees of designer control and to achieve different goals.

Keywords: Pokémon, Procedural Content Generation, Game Aesthetics, Decision Trees, AI-assisted game design

1 Introduction

As a practice, both game design and artificial intelligence (AI) tend to treat digital games as *systems* which, while not as strictly mathematical as game theory [1], prioritize the discovery of winning strategies. For game design, such winning strategies must ideally be varied to avoid a "shortcut" to victory; for AI aiming at efficient game playing, the task revolves around discovering such winning strategies [2] (and even exploiting shortcuts). For similar reasons, research in procedural content generation in games largely targets creating playable, balanced content [3]; at best, the player's experience is accounted for via computational models [4] which are again based on interactions with the system (such as jumps or enemy hits). However, an important part of play goes beyond the game's mechanics (or their combination) but expands into the aesthetic experience [5] of exploring a vast and colorful world, getting involved in the backstories of non-player characters and listening to the audioscapes formed by the soundtrack and environmental sounds based on the player's location.

It is therefore important that elements other than the game's mechanics or the game levels' architecture are considered, both from the game design and from the AI perspective. For game design, adding visual cues in a vast open

world via e.g. landmarks can help player navigation by directing them towards important areas [6]. Similarly, rendering a weapon or an enemy in a way that makes their mode of interaction or challenge level recognizable by mere visual inspection makes for an intuitive player interaction. For AI, building models of player experience based on audiovisual stimuli can complement player modeling based on in-game events [7]. For procedural generation, identifying patterns of visual appeal or associations between a shape or color and its in-game purpose allows us to generate visual depictions of content in an informed way.

This paper describes how a mapping between visual representation and in-game identity can be learned from simple image information, and then used to drive the generation of new game visuals. Identity in games is particularly important, especially concerning player avatars; the mapping between an avatar's visual identity and user-provided game statistics or narrative has been explored in [8]. In this paper, we instead focus on non-player characters, i.e. on the pet creatures of the Pokémon series; moreover, we learn the design principles and rules behind the choice of colors to identify and represent specific elements (e.g. fire, water) or other Pokémon types (e.g. fighting, psychic). There are many goals and directions for this research: on the one hand, the learned models can inform game designers on the principles behind artists' renderings and guide them to create new Pokémon; on the other hand, the learned model can be used to predict the type of unseen Pokémon sprites. Taking advantage of the latter, this paper shows how evolution can create new Pokémon sprites by changing the color mapping of existing ones. This can be used to assist designers or to automate the design of recolored Pokémon with new types, similar to *Alolan* Pokémon or more generally *shiny* Pokémon in existing games. As demonstrated in Section 5, this can be applied to change the type of a single Pokémon to a designer-defined one, to change the type of a set of Pokémon of the same type to any other, or to balance the instances of each Pokémon type. However, far more applications for the learned model of visual identity are discussed in Section 6.

2 Related Work

While not prevalent in AI for games, creating computational models of visual identity is a core direction of computer vision for tasks such as object detection [9] and image recognition [10]. Just as computer vision tasks try to answer "what are the essential visual clues of a chair?", this research attempts to answer "what are the essential visual clues of a fire Pokémon?". Computer vision models have also been used for generation of new visual artifacts that match real-world knowledge, such as alphabets based on Optical Character Recognition accuracy [11] or 3D models based on the confidence of a deep learned image recognition system [12]. In games, computer vision has been applied to recombine facial features of 2D avatars in a way that makes them recognizable as faces of celebrities [13].

While the design of computer games hinges on a number of creative facets [14], research in AI and content generation focuses heavily on the more "measurable" rule and level design tasks. For generating in-game visuals, evaluation

Table 1. Types of Pokémon and the number of instances of this type in the database (including dual types).

Water (141)	Normal (116)	Flying (113)	Grass (109)	Psychic (100)	Bug (83)
Ground (75)	Fire (72)	Poison (69)	Rock (67)	Fighting (63)	Dark (60)
Electric (60)	Dragon (59)	Steel (58)	Ghost (55)	Fairy (53)	Ice (43)

Fig. 1. Pokémon sprites which capture extremes of saturation, sprite size, color etc.

often relies on human feedback via interactive evolution [15, 16], or an inferred [17] or stated [18] objective of the designer. Human expertise can also be used indirectly, however, to learn and replicate patterns of e.g. level design from high-quality human designs [19] or, as in this study, to learn the rules behind visual associations inserted by a game's designers and exploit them to generate new content (via their color recombinations) which still retain their human-provided form (via the sprite's shape and brightness information).

3 Processing the Pokémon Dataset

Pokémon are fantastical creatures featuring in the Pokémon game series by Nintendo from 1996 to 2016. The games revolve around capturing and using Pokémon in combat against other Pokémon. Each Pokémon has one or two *types* (e.g. fire Pokémon or grass–poison Pokémon), which affects their moves (which also have a type) in combat. Moves that cause damage have their damage amplified depending on its type and the type of the enemy Pokémon: e.g. a fire move deals double damage to ice Pokémon, but half damage to fire Pokémon. A Pokémon's type thus affects both its defenses (versus moves of different types) and its offenses (its available moves and their types).

3.1 The Dataset

The Pokémon dataset was collected from the Pokémon Database[1] which contains statistics and sprites of every Pokémon from all seven generations of the Pokémon game series. While Pokémon statistics include numerical data such as health, attack, defense etc., the Pokémon's type is very important as choosing the right Pokémon to counter another Pokémon's type is the key to victory. Each Pokémon also has a low-resolution sprite for its visual representation.

The main focus of this paper is to identify the relationship between the visual depiction of different Pokémon, especially regarding their color, and their

[1] `https://pokemondb.net/pokedex/all`

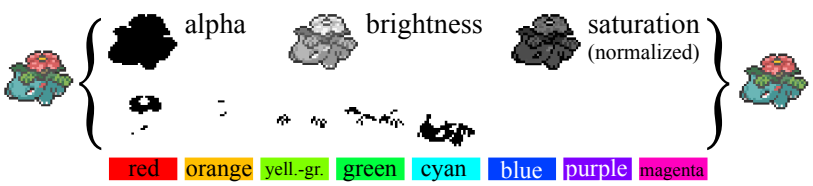

Fig. 2. Breakdown of the base sprite (far leftt) into alpha, brightness, saturation, and color channels for the 8-color encoding. The channel information can be used to reconstruct the sprite (far right) with some color accuracy loss.

type(s). The dataset contains 908 Pokémon entries, including evolved forms and alternate forms. The dataset includes 18 types of Pokémon shown in Table 1; out of those, Water, Fire and Grass are considered the three basic elemental types. Table 1 includes the number of instances of each type in the database; since 488 Pokémon have dual types, the total instances of type are 1396. Water Pokémon are by far the most common; ghost, fairy and ice Pokémon are far fewer.

3.2 Decomposing Pokémon Sprites

Each Pokémon sprite is 40 pixels wide and 30 pixels tall: the actual Pokémon is much smaller (leaving the rest of the canvas transparent). For the purposes of this paper, the image is split into constituent parts (channels) which can be used to reconstruct it. In the Hue, Saturation, Brightness (HSB) image format, the three channels describe the type of color of each pixel, how vibrant the color is, and how bright or dark it is, respectively; an additional Alpha channel describes whether the pixel is transparent. The HSB format is followed here, with some important modifications. Each image is split into the alpha channel (black pixels for opaque color and white pixels for empty areas), the brightness channel (i.e. a grayscale version of the image) and the saturation channel which is multiplied, in this case, by the brightness channel since dark pixels (low brightness) are not perceived as saturated by the human eye regardless of saturation score. Saturation in this paper always refers to this normalized saturation channel which also performed well in [8]. The hue channel is problematic as it is a wheel: high and low hue scores are both red-tinted. This paper splits hue into value ranges (colors) and stores them as black and white images representing presence or absence of that color. Therefore, all red or almost-red pixels are black in the red color channel, blue or almost-blue pixels are black in the blue color channel etc. (see Fig. 2). To avoid noise due to black or white pixels being stored in a random color channel, sprite's pixels with less than 5% in their normalized saturation value (i.e. low saturation, low brightness, or both) are omitted from all color channels.

These channels are then processed to derive visual metrics for each Pokémon sprite. This paper uses only the color ratios as a metric, calculated as the ratio of black pixels in a color channel (see Fig. 2) over the Pokémon's total number of pixels (i.e. black pixels in the alpha channel). Since the sprite contains black pixels (e.g. as outline) or white, the sum of color ratios is usually far below 100%.

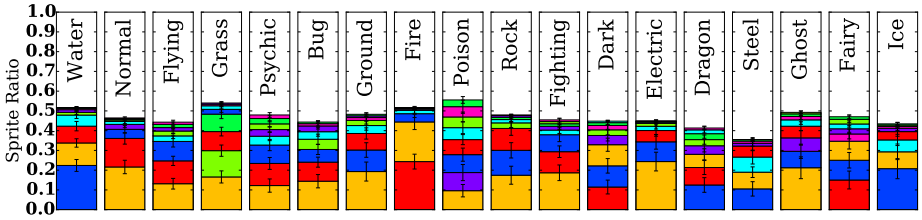

Fig. 3. Color ratios per Pokémon type. Error bars display the 95% confidence interval of the average ratios shown.

3.3 Analysis of Pokémon Sprite Metrics

Pokémon sprites are very diverse in terms of sprite size, brightness as well as saturation; sprite sizes range from 92 to 585 pixels (with an average of 209). Out of those pixels, 48% are occupied by any color on average. For this analysis, the color channels considered are set to 8 (see Fig. 2). Across all sprites, the most prominent color is orange (average ratio of 15%), red (11%) and blue (10%).

Looking at specific types of Pokémon, Fig. 3 shows the distribution of colors in their sprites (sorted by coverage ratio). Most Pokémon types have a large ratio of orange pixels, but there are obvious differences among different types of Pokémon. Fire Pokémon have a high ratio of red and orange pixels and little else, while poison Pokémon have a much fairer distribution of colors. Water and ice Pokémon have the highest coverage in blue pixels, while only poison and ghost Pokémon have a substantial presence of purple colors. Grass Pokémon have more coverage in green or yellow-green colors than other types; steel Pokémon have the least coverage in any color. It is expected that most of these patterns can be learned by the classifiers discussed below.

4 Building a Classifier for Pokémon Types

In order to understand how color reflects a Pokémon's type, a machine learning approach is used to predict unseen Pokémon such as those evolved in Section 5. Each Pokémon belongs to one or two types, so the learning task is one of *multi-label classification* since types are not mutually exclusive. In this paper, decision trees are used to classify Pokémon types, based on their image properties (i.e. image metrics) described in Section 3.2. Decision trees were chosen as they are among the few algorithms that handle multi-label classification tasks out-of-the-box, but also due to the fact that they are human interpretable and can be visualized. As a white box model, a decision tree allows a designer to understand the rules that make a Pokémon visually reminiscent of its type. The inputs of the decision tree are exclusively the sprite's image metrics, i.e. the different color ratios and, optionally, the sprite size, average and standard deviation of brightness, average and standard deviation of saturation. The output is an array of 18 integers each representing membership in a type (outputs can be 0 or 1).

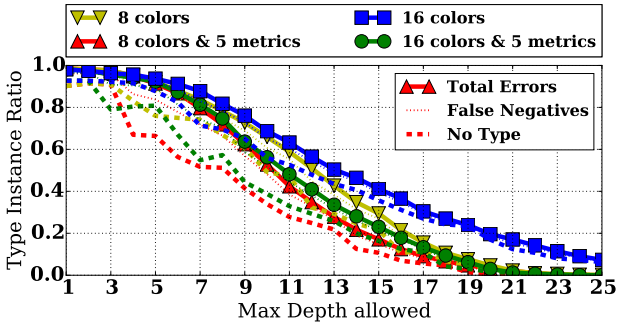

Fig. 4. Sensitivity of the decision tree classification task for different maximum tree depths and inputs. Type errors are normalized to the 1396 instances of type and Pokémon with no type are normalized to the 908 Pokémon in the dataset.

All 908 data points are used to build the decision tree based on the ground truth of the Pokémons' types. Since decision trees may suffer from over-fitting, it is worthwhile to perform a sensitivity analysis regarding the maximum depth of the tree. The larger and more complex the decision tree, the more likely it is that niche rules were added that do not generalize to unseen Pokémon sprites. It should be noted that for all intents and purposes all Pokémon from the games will technically be "seen" by the classifier; however, as we intend to create new sprites based on recoloring methods of Section 5, its ability to generalize remains important. Moreover, alternative inputs could also be considered such as more color channels or including the 5 non-color metrics. The sensitivity analysis below will explore the impact of all those parameters. The core performance metric is of course the accuracy of the classifier, i.e. the sum of false positives (i.e. the classifier predicts that the Pokémon belongs to a type, but in truth it does not) and false negatives (i.e. the classifier fails to predict one of the Pokémon's types). Due to the nature of the dataset and the purposes of this study, an important secondary performance metric is the number of Pokémon which have no type whatsoever (i.e. 0 in all the classifier's outputs); Pokémon that can not be classified (not even mis-classified) are problematic for the purposes of evolving a new color palette, as the overall number of Pokémon essentially becomes smaller.

Fig. 4 shows how accuracy and the number of Pokémon without types change as the depth of the decision tree increases. Using only color information (either 8 or 16 color channels) needs larger trees for accurate classification despite the fact that the metrics are fewer. Including the 5 other image metrics described above helps with classification tasks, and using only 8 colors seems beneficial as errors and Pokémon without a type drop more abruptly in larger trees. While the number of errors is fairly low for the 13 feature input set after a depth of 15 and for the 8 colors after a depth of 17, the number of Pokémon without a type is a concern. Erring on the side of caution, the ratio of Pokémon without a type drops to 0.5% at depth 20 for the 13 feature set and at depth 22 for the 8 colors; their misclassificiation ratio is 2% and 0.6% respectively. While the 13 feature set

is beneficial as it takes into account information missing from the color channels (including white and black pixes or how saturated the colors are), over 58% of its decision nodes test those 5 non-color features. When evolving a color palette in Section 5, the system can only control the colors (and not e.g. the size of the sprite) which causes some Pokémon with specific sizes or brightness values to never be able to change their type. For that reason, and despite somewhat lower accuracies, the decision tree using 8 colors as input (and a depth of 22) is used in the remaining experiments of this paper.

5 Evolving the Pokémon Pallette

The classifier of Pokémon types based on their color ratio can be used to analyze existing Pokémon and could be useful to designers as a white-box model. However, in this study it serves an ulterior goal of classifying unseen procedurally generated Pokémon sprites. In this paper, the generative process revolves around the recoloring of existing Pokémon sprites from the database of 908 sprites, achieving new color combinations without changing the sprite's outline or the Pokémon's physiology. This design choice was taken not only to test how much can be achieved with a simple representation, but primarily to ensure that recognizable and high-quality Pokémon can be produced. Moreover, the impact of color on people's perception has been studied extensively in advertising or psychology but rarely in game design (and even less so in procedural content generation); therefore evaluating how generated color palettes can be used to strengthen human assumptions on the meaning of color is particularly relevant.

As described in Section 3.2 and Fig. 2, each Pokémon sprite is split into different image channels. These images can be recombined to create a sprite almost identical to the original. As color channels are binary and the sprite is colored based on the midpoint of the hue range, some errors in the colors of reconstructed sprites may occur; with 8 colors, however, obvious miscolorations are rare and only in some Pokémon. This method of reconstructing a Pokémon sprite from its alpha, brightness, saturation and color channels will be used to re-color existing Pokémon sprites via the process explained in Fig. 5. The top row shows the 8 color channels (which are binary as per Fig. 2) and the color palette mapped to them: those are colored with the appropriate hue as shown in Fig. 5 and combined with the brightness and saturation channels to reconstruct the Pokémon on the top right. However, if these same color channels are mapped to different hues as shown in the bottom row, when recombined they will create a very differently colored Pokémon which remains recognizable due to the alpha, brightness and saturation channels.

In order to generate new color palettes, the original 8 colors are taken and swapped randomly. As we avoid having two channels map to the same color in this study, the possible permutations of 8 colors is $40,320$; searching this space exhaustively would be cumbersome. Instead, evolution is used to stochastically search the space of color combinations for ones that satisfy the designer's objectives. These objectives will be discussed in the next subsections. Regardless of

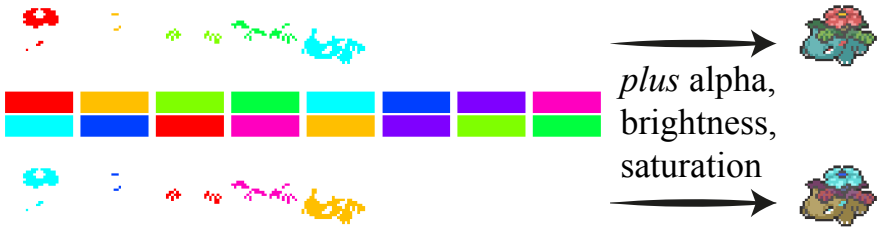

Fig. 5. Recoloring process with a re-shuffled color palette (bottom).

objective, the evolutionary process is the same: an initial population of random permutations of the colors' original order evolves via mutation alone to increase an objective score. Crossover is not used as it could lead to multiple channels with the same color, i.e. mostly monochrome sprites. Mutation can (a) create a new permutation (ignoring the previous genetic encoding), (b) reverse the order in the gene, (c) swap a random color with another random color in the palette's order, (d) swap a random color with its adjacent in the palette, (e) shift all colors to the left (red to magenta etc.) or (f) right (green to cyan etc.). One of these mutations is applied to each individual, and each mutation has an equal chance of being chosen. In the order they are presented, mutations range from disruptive to minor changes; their combination should be sufficient to drive evolution towards more promising solutions without risking genetic drift. Evolution follows a μ, λ evolutionary strategy [20] with 50% elitism: in each generation the fittest half of the population is copied without mutation, and these elites are also chosen (at random) to produce offspring via mutation. The least fit 50% of the individuals are replaced by the mutated offspring of the most fit 50%. This is a somewhat aggressive evolutionary strategy (as it lacks the stochasticity of roulette wheel selection and its elitism ratio is high), which is why mutations that can introduce new genes are desirable to avoid premature convergence.

In the following subsections, a number of experiments will cover different objectives of recoloring Pokémon sprites, from local changes to sweeping changes on the entire dataset. All objective functions in the following experiments must be maximized during evolution, and there are criteria for early stopping if the desired value (which is known in advance in all the listed cases) is reached. Finally, in order to avoid creating color combinations that can not be handled by the classifier, all fitness scores are reduced by the penalty function P of eq. (1) which deters more Pokémon of no type than in the original dataset, as well as Pokémon which are predicted to have more than two types.

$$P = C_e min(0, E - E_{orig}) + C_{mt} \sum_{i \in S} \left(\sum_{t \in T} p_t(i) - 2 \right) \quad (1)$$

where T is the set of all 18 types; $p_t(i)$ is the predicted value (0 or 1) for type t of the decision tree using the evolving pallette of i as input; E is the current number of Pokémon with no type and E_{orig} is the number of Pokémon with no

Fig. 6. Evolving a pallette for changing one Pokémon's types. In the first row, Bulbasaur (1st) is evolved to gain the fire type (2nd) and the water type (3rd). In the second row, Charmander (1st) is evolved to gain the grass type (2nd) and the water type (3rd). In the third row, Squirtle (1st) is evolved to gain the fire type (2nd) and the grass type (3rd).

type before recoloring for the same dataset S; C_e and C_{mt} are constants (in this paper $C_e = C_{mt} = 10$).

5.1 Customizing a single Pokémon

The simplest use of a generative color palette is to change a single Pokémon sprite to a desired new type. This is obviously a local change which only affects the Pokémon sprite in question, but can be used to create a broad range of Pokémon (especially considering dual types) from a single sprite. The decision tree is used to classify the recolored Pokémon sprite based on the evolving color palette; its fitness, as shown in eq. (2), is proportional to the misclassifications from its original type(s) and its correct classifications of the desired type(s).

$$f_1 = \frac{1}{|D|}\sum_{t\in D} p_t(i) - \frac{1}{|O|}\sum_{t\in O} p_t(i) - P \tag{2}$$

where O and D are the set of types of the original Pokémon i and the desired Pokémon i respectively. P and all other notations are covered by eq. (1).

As a demonstration, the three basic elemental types (water, fire and grass) will be used to recolor the starter Pokémon of the first generation of games. The three starter Pokémon shown in Fig. 6 are widely recognizable: the grass-poison Pokémon is Bulbasaur, the fire Pokémon is Charmander and the water Pokémon is Squirtle. Evolution will recolor them so that they no longer belong to their original types but instead belong to one of the other elements of that triangle.

Table 2. Ratio of Pokémon retaining their original type at the end of evolution which attempts to remove the type from a subset of the database. In the "Replaced by" row, the table includes the most popular type among the recolored Pokémon of that subset. Results are averaged from 10 independent runs.

Type	Water	Normal	Flying	Grass	Psychic	Bug	Ground	Fire	Poison
Remaining	0.07%	0%	0.8%	0%	2%	0%	0%	0%	2.03%
Replaced by	Grass	Psychic	Psychic	Ice	Grass	Fairy	Psychic	Poison	Psychic
Type	Rock	Fighting	Dark	Electric	Dragon	Steel	Ghost	Fairy	Ice
Remaining	0%	0%	0%	0%	1.19%	2.41%	0%	0%	0%
Replaced by	Grass	Psychic	Dragon	Steel	Fairy	Dragon	Grass	Ground	Fairy

Results in Fig. 6 are chosen among 10 evolutionary runs for each objective, with 20 individuals evolving for 50 generations (although evolution terminated prematurely in all runs). Evolution evidently took advantage of the learned pattern that water Pokémon tend to be blue, fire Pokémon tend to be red and grass Pokémon tend to be green (see Fig. 3). The second type of the recolored sprites is more insteresting, as purples in the recolored fire Squirtle and the water Bulbasaur give them the ghost type. Although this is a simple task that could likely be solved by random swaps and/or a hill climber, evolution easily found these results which is promising for the more challenging tasks of Sections 5.2 and 5.3.

5.2 Removing a Pokémon type

As shown in Section 5.1, it is possible to recolor a single Pokémon in a way that it is not recognized as belonging to its original type. Taking this approach one step further, an entire category of Pokémon sprites can be recolored in order to change their type. In the experiments of this section, we focus on a set of Pokémon grouped by type and attempt to recolor them in a way that removes that specific type. Contrary to the previous approach, evolution now operates on a larger scale as it needs to find a pallette that works on a large set of Pokémon; however, the goal of evolution is only to make these sprites not be identifiable as Pokémon of their original type but does not include a designer-specified target type. This allows more freedom for initially compatible Pokémon (in type, if not strictly visually) to adopt different appearances and types. The system extracts a set S_t of all Pokémon belonging to type t (including dual type Pokémon with t as one of their types) and attempts to maximize the fitness of eq. (3).

$$f_2 = -\frac{1}{|S_t|}\sum_{i \in S_t} p_t(i) - P \tag{3}$$

where i the Pokémon in set S_t which belonged, originally, to the same type t; P and other notations are covered by eq. (1).

Table 2 shows how removing the type of specific sets of Pokémon is handled by the evolutionary system in terms of both the fitness (i.e. the ratio of Pokémon that retain their type) and the most popular type that replaces it. All results

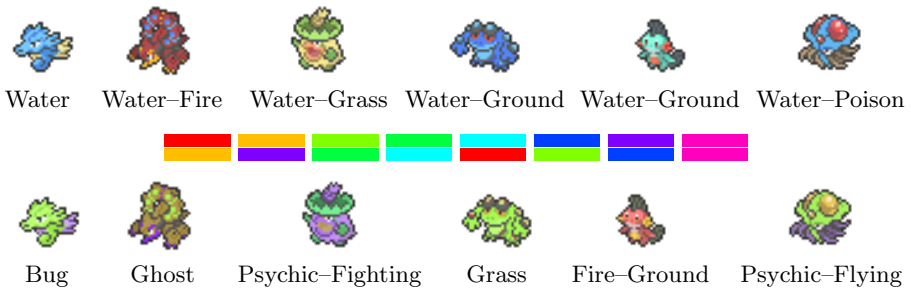

Fig. 7. A sample of water Pokémon's which, after evolution, have been converted to a diverse set of types excluding water.

are averaged from 10 independent evolutionary runs, collected after evolution is carried out on 20 individuals for 50 generations. Most evolutionary runs succeed in completely removing the chosen type, and only fire and dragon types had one Pokémon of that type remaining even in their most successful run. The easiest types to eradicate are ghost and ice, which are fully removed in less than 3 generations on average. Looking at the most popular types found in the recolored Pokémon of different subsets, the psychic type is surprisingly popular and the water type surprisingly absent as a prevalent replacement. As will be discussed below, this is likely because psychic Pokémon often feature cyan or purple hues (see Fig. 3) which are rarely shared by other Pokémon. The simplest way to remove a Pokémon's original type is thus to recolor it in such rare hues.

Among the most successful attempts at changing the type of an entire set of Pokémon is shown in Fig. 7. The shown evolved color pallette managed to remove the water type from all 141 water Pokémon; water was the most popular type based on Table 1. While none of the recolored Pokémon were classified as belonging to the water type, all other types are represented. The most popular type among recolored Pokémon is grass (28%) and flying (25%) and the least popular is fairy and poison (1% of each). The examples of Fig. 7 shows that blue sprites became yellow-green and orange sprites became purple; generally, colors originally adjacent now have stark contrasts. Since many water Pokémon were initially blue, it makes sense that more grass Pokémon are predicted with the evolved mapping of blue to green (see Fig. 3).

5.3 Balancing the number of Pokémon per type

The most ambitious application of the proposed approach is to apply the recoloring rules globally, to the entire database. There are many reasons for doing so, including those motivating evolution in the previous sections. For instance, evolution can recolor the entire database to minimize the number of Pokémon of a specific type, as in Section 5.2 (e.g. if a designer decides to eradicate one specific type from the lore) or to create as many Pokémon of a specific designer-defined type (or combination of types) as in Section 5.1. In order to show another

Fig. 8. Sample sprites evolved to balance the ratio of Pokémon types.

application of the proposed approach, this section instead attempts to create recoloring rules which minimize the discrepancy between the number of Pokémon of each type. As seen in Table 1, some types are present in more Pokémon than others (the number of water Pokémon is triple that of ice Pokémon). Using the entire dataset of 908 sprites, evolution attempts to maximize f_3 of eq. (4), which is the standard deviation of the number of Pokémon per predicted type.

$$f_3 = -\sqrt{\sum_{t \in T}(N_t - \bar{N})} - P \tag{4}$$

where $N_t = \sum_{i \in S} p_t(i)$, i.e. the number of Pokémon predicted to belong to type t; $\bar{N}$ is the average of all N_t for $t \in T$; T is the set of all 18 types and S is the set of all 908 Pokémon. P is formulated in eq. (1).

Results are collected from 10 runs where 20 individuals evolved for 50 generations. Optimization was generally consistent, but this section analyses in depth the fittest result among those runs. The fittest color pallette is shown in Fig. 8: it maintains the original color of primary, popular colors (red, orange, green, blue) while altering the rest. Therefore, many of the Pokémon sprites do not change color or type, e.g. the fire Pokémon of Fig. 8. Unlike results in Fig. 7, evolution here follows a more conservative recoloring strategy.

Since many of the colors remain the same after evolution, it should not be surprising that the color ratios of different types largely stay the same. Figure 9 shows the color ratios for different Pokémon types (as predicted by the classifier) with the evolved palette. General patterns remain the same: water Pokémon tend to be blue, fire Pokémon tend to be red, grass Pokémon tend to be green, and all Pokémon have a fairly high ratio in orange. There are notable differences too: magenta is much more prevalent (it has the highest ratio for poison Pokémon now), while cyan is rarer in water and ice Pokémon compared to Fig. 3.

The distribution of predicted types with the new palette is shown in Table 3, sorted by popularity. Notable changes from Table 1 is the drop of water and grass Pokémon by over 20 instances (slightly less for dragon and fire). At the same time, most other types of Pokémon increase, especially fairy and ghost Pokémon.

Table 3. Predicted types in the recolored sprites that maximize f_3.

Normal (120)	Water (116)	Flying (110)	Psychic (105)	Bug (88)	Fairy (87)
Grass (81)	Rock (78)	Poison (70)	Ground (69)	Electric (69)	Ghost (68)
Fire (66)	Dark (61)	Fighting (59)	Steel (56)	Dragon (49)	Ice (42)

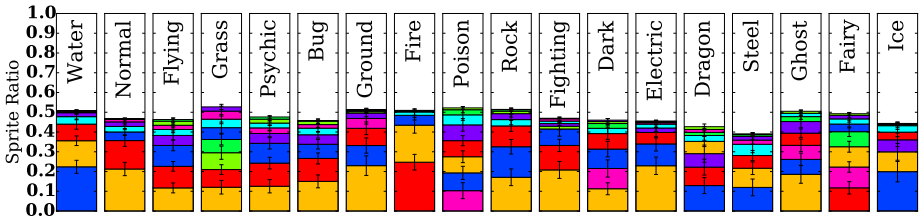

Fig. 9. Color ratios per Pokémon type of the evolved set that maximizes f_3. Error bars represent the 95% confidence interval of the displayed average ratio (per type).

This shouldn't be surprising, as the more prevalent magenta and purple colors were quite prevalent in the original fairy and ghost Pokémon respectively, based on Fig. 3. While the distribution of types is still not entirely balanced, there are more types with values closer together by reducing the number of some popular types (water, grass) and increasing that of less popular ones (ghost, fairy). The deviation of instances among the types was 27 in the original dataset; with the evolved palette it is 22. This is admittedly not a large improvement, but it is likely the limit of what can be achieved with a simple color swap and without introducing more Pokémon that are classified as having no type.

Observing the sample Pokémon of Fig. 8 shows some trends of the evolved color mapping. For many sprites, due to colors such as yellow and red remaining the same, their appearance and type was retained (e.g. the fire Pokémon); even when recolored, some sprites retained their type e.g. the psychic–fairy Pokémon. On the other hand, even small changes in color can result in changes of types as in the water–grass Pokémon which gains two completely new types due to a change in the color of the "hat" and limbs. Of interest are the two grass–poison Pokémon which, while one is an evolution of the other, when recolored gain different colors (one is originally more cyan than the other) and different types.

6 Discussion

The purpose of experiments in this paper was to explore and demonstrate how the proposed evolutionary system can be used to create new Pokémon by recombining their color palettes. As a small sample of the possible use cases of this approach, the recolored palette can change the type of a Pokémon to a desired type, for local changes, remove a type from all Pokémon of that type, and find a new color palette for all Pokémon to change the distribution of their types. Results show that all of these are largely successful: in most cases evolution is

able to perform its goal (find all desired types, remove all instances of a type from a subset of the database) although for the more challenging task of global palette swaps, it seems that it requires that certain colors retain their original hues and consequently many of the Pokémon retain their original type.

The decision tree classifier showed that at sufficient depth it can be accurate in its predictions. Moreover, as a white-box system it can be a useful design tool, especially if some of its rules are converted into natural language: e.g. "if sprites are large and their red color is higher than average. . . ". There is a concern regarding the number of instances which could not be classified (i.e. Pokémon without a type) which led to adopting a large tree that perhaps overfits to the data. Experiments on evolving a color palette used a tree with only decision nodes based on color ratios; this was less accurate overall to a model which includes information on brightness, sprite size and saturation. Preliminary experiments using those additional features during evolution were problematic as some Pokémon would not change their type due to metrics beyond the control of evolution (e.g. small Pokémon were overfitted to always be bug types). While using additional metrics is valuable as a design tool and for designer feedback, when used for automated generation via recoloring it underperforms.

Admittedly, both the motivation and the evaluation of this paper was based on designer intuition and the desire to only partially automate the design process. For instance, there is no parameter tuning reported in the results, and parameters such as mutation and population size are not argued for. The purpose of experiments was to demonstrate rather than benchmark the potential of the algorithm; while many more experiments with different fitness functions, population sizes etc. were performed, only a handful are reported to give more of a "taste" of potential applications. For instance, experiments with penalty functions which reward fewer Pokémon with no predicted type showed that this reward dominated the fitness in the more challenging tasks of Section 5.3. Using crossover resulted in many colors in the original sprite mapped to the same color in the recolored version, reducing the visual appeal while also not reaching higher fitnesses than mutation alone. In terms of reporting, there are not many baselines that could be considered for such a task (apart from exhaustive search); significance testing in an approach aiming at recoloring Pokémon in interesting ways seems exorbitant, considering that most evolutionary runs ended prematurely as they reached the stated objective. As a final note, the use of existing color channels, and their recoloration in a one-to-one mapping was a design decision in order to increase the quality of resulting sprites. Changes in saturation values or brightness channels is also possible, provided there is some constraint that deters the creation of "pure noise"; in that case, computer vision approaches such as deep learning would be more suitable than the numerical representation of the image used here as input to the classifier.

Finally, Pokémon generation could have more dimensions than their sprites and type mapping. Pokémon statistics such as hit points and attack could be classified, along with their type, based on image metrics; this could allow a more fine-grained sprite generator to create Pokémon for specific strategies (e.g. a

"Wall" Pokémon with high hit points and high defense). Moreover, the moves of the Pokémon could be mapped to their sprite's appearance and type (and possibly stats). Consistency of Pokémon types in their evolved forms, or in higher resolution versions could also be considered. More niche computational models such as a name generator learned from current Pokémon names could also potentially be used to give names to the newly generated Pokémon sprites to match their new types. There are many straightforward as well as intricate ways in which this initial study can be expanded.

7 Conclusion

This paper introduced a system for decomposing Pokémon sprites into image metrics that can be used to learn a mapping between the Pokémon's type(s) and its visual appearance. Decision trees were shown to be fairly accurate at sufficient depth, especially when combining color information with other information such as sprite size and brightness. Using an evolutionary algorithm, color palettes based on the original sprites and colors were evolved to create new Pokémon on a local or global scale that matched to or diverged from specific types, as well as to lessen imbalances between the instances of each type. Results showed that at its core the recoloring strategy is able to perform most of these tasks, creating in most cases visually coherent and appealing Pokémon assigned to new types. This initial study could be expanded with more features to learn (either based on color, on other visual metrics, or on in-game statistics) and to generate. Several directions for future work must be explored, such as the use of a more granular representation, different machine learning approaches, and the generation of more Pokémon details beyond their spites.

Acknowledgements

Pokémon images and names are copyright of Nintendo/Game Freak; no copyright infringement is intended. No monetary profit was made from this article.

References

1. von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press (1944)
2. Perez-Liebana, D., Samothrakis, S., Togelius, J., Lucas, S.M., Schaul, T.: General Video Game AI: Competition, challenges and opportunities. In: Proceedings of the AAAI Conference on Artificial Intelligence (2016)
3. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games 2(1), 1–16 (2010)
4. Yannakakis, G.N., Togelius, J.: Experience-driven procedural content generation. IEEE Transactions on Affective Computing 99 (2011)
5. Hunicke, R., Leblanc, M., Zubek, R.: MDA: A formal approach to game design and game research. In: Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence (2004)

6. Winters, G.J., Zhu, J.: Guiding players through structural composition patterns in 3d adventure games. In: Proceedings of the Foundations of Digital Games Conference (2014)
7. Yannakakis, G.N., Spronck, P., Loiacono, D., Andre, E.: Player modeling. In: Dagstuhl Seminar on Artificial and Computational Intelligence in Games (2013)
8. Lim, C.U., Liapis, A., Harrell, D.F.: Discovering social and aesthetic categories of avatars: A bottom-up artificial intelligence approach using image clustering. In: Proceedings of the International Joint Conference of DiGRA and FDG (2016)
9. Basri, R., Costa, L., Geiger, D., Jacobs, D.: Determining the similarity of deformable shapes. Vision Research (1998)
10. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision 115(3), 211–252 (2015)
11. Martins, T., Correia, J., Costa, E., Machado, P.: Evotype: Evolutionary type design. In: Proceedings of the International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART), pp. 136–147. Springer International Publishing (2015)
12. Lehman, J., Risi, S., Clune, J.: Creative generation of 3d objects with deep learning and innovation engines. In: Proceedings of the International Conference on Computational Creativity (2016)
13. Kao, D., Harrell, D.F.: Exigent: An automatic avatar generation system. In: Proceedings of the Foundations of Digital Games Conference (2015)
14. Liapis, A., Yannakakis, G.N., Togelius, J.: Computational game creativity. In: Proceedings of the Fifth International Conference on Computational Creativity (2014)
15. Risi, S., Lehman, J., D'Ambrosio, D.B., Hall, R., Stanley, K.O.: Combining search-based procedural content generation and social gaming in the petalz video game. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (2012)
16. Hastings, E.J., Guha, R.K., Stanley, K.O.: Evolving content in the galactic arms race video game. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2009)
17. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games 4(3), 213–228 (2012)
18. Howlett, A., Colton, S., Browne, C.: Evolving pixel shaders for the prototype video game subversion. In: Proceedings of the AI and Games Symposium (AISB'10) (2010)
19. Summerville, A., Mateas, M.: Sampling hyrule: Multi-technique probabilistic level generation for action role playing games. In: Proceedings of the Foundations of Digital Games Conference (2015)
20. Beyer, H.G., Schwefel, H.P.: Evolution strategies – a comprehensive introduction. Natural Computing 1(1), 3–52 (2002)