Orange Peel Optimization

David Swart
Waterloo, Ontario, Canada; dmswart1@gmail.com

Abstract

The woodcut prints *Frog* and *Crow* have the special property that one could cut each figure out and, after tucking in some feathers or legs, roll them up into a paper-craft globe. These designs came about by mathematically peeling spherical imagery like an orange and seeing which animals the resulting shapes looked like. This paper describes how to reverse this design process. We describe a method to start with the desired 2D shape and determine how to unpeel an orange to match it. We then explore some creative possibilities of this new method, making new woodcut designs and even creating a font.

Introduction

Mathematical functions that map the points of a sphere to points on a plane are useful for creating world maps. It is what cartographers do. But spherical maps are also an opportunity for many kinds of creativity. For instance, an artist can start with spherical imagery (either an all-around panorama, or the globe itself) and achieve interesting results by projecting the imagery to a plane in many interesting ways. Germán et al. [1] described early artistic projections of spherical panoramas. Lee [4] is an artist that showcases beautiful photo-sphere projections. And making art from spherical imagery has been a recurring theme in my own projects [7-10].

The most accessible way for anybody to creatively explore this kind of fun is by peeling orange peels. (Throughout this paper *orange peels* can be taken to mean any citrus peels, but especially mandarins). It is common to peel an orange in one piece, flatten it on a table and see what pattern it makes (such as flower petal designs or spiral shapes). The Japanese artist Yoshihiro Okada [6] is the foremost master of this art form and has made dozens of very impressive animal shapes out of orange peels. Following in his footsteps, Liu et al. [5] developed a computer program that allows users to interactively find very high quality orange peel designs.

My own motivation to explore orange peel art began with the challenges I faced designing the two woodcut prints *Frog* and *Crow* shown in Figure 1. The design for *Frog* began by unwrapping a sphere in a geometric way: with three sinusoidal segments radiating from one pole. To me, it looked like a frog, so I enhanced the design with some froglike features. Together with Evan Swart, we chose to produce it as a woodcut print to emulate the graphic artist M. C. Escher in our own small way. To continue the series, we tried to deliberately create an animal shape by hand. At first we intended to make a spider shape which ended up looking like a lobster. When we tried to refine the lobster shape it morphed again into a bird. Admitting defeat, we accepted the shape as it was, added feathers and a beak and produced the second print *Crow*. Our long term goal is to continue to make one print for each of five classes of vertebrates: fish, amphibian, reptile, bird, and mammal. To achieve this goal, we needed a way to deliberately unwrap a sphere into a shape we want.

In this paper we describe a way to optimize an orange peel into a shape that is as close as possible to a given target silhouette. The next section reviews some previous work onto which the new optimization method builds. Then we describe exactly how to cast our goals as an optimization problem, and then fill in more details about the publicly available software implementation. Finally, we discuss some new art projects that showcase the new capabilities including the remaining designs of our animal woodcut series.

Figure 1: *The woodcut prints* Frog *and* Crow *(42cm×30cm) made from two blocks each by David and Evan Swart. Each figure can be cut out and rolled up into a 6cm radius globe.*

Turtles and Skeletons

In a paper from Bridges 2009 [8], I describe a computer program that takes a user defined scheme and performs a mapping from the surface of a sphere to the plane. The user starts by inputting a set of line segments (great circle arcs) on the sphere called a *skeleton*. They do so by using a logo-like turtle language with commands such as 'move forward', 'rotate left', 'draw a line' etc. Two turtles, one on the sphere and one on the plane simultaneously trace out corresponding skeletons in both spaces. The program uses these skeletons to unwrap the surface of a sphere onto the plane. It takes the Voronoi regions of each of the line segments on the sphere (that is, the sets of points closest to each line segment on the sphere) and then maps them to a region surrounding the corresponding line segment on the plane. Figure 2 contains a diagram of the process using a tennis ball design as an example.

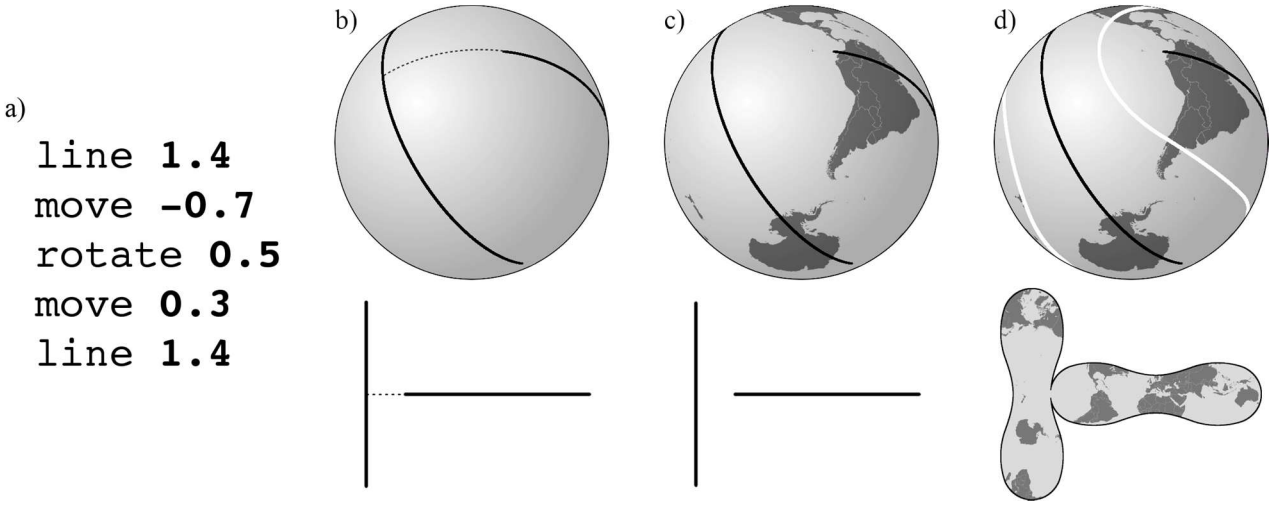

Figure 2: *A single turtle program (a) directs two turtles to draw two paths (b), one on a sphere and one on a plane defining two skeletons (c) made of corresponding line segments. The regions of points nearest to each line segment on the sphere are mapped (d) to the corresponding segment on the plane.*

The resulting program is a versatile tool that allows for a hands-on method to creatively explore many map projections including the designs for *Frog* and *Crow*. Importantly, this program has a small set of numerical values corresponding to the angles and lengths of the skeleton (shown in bold in Figure 2a). These numerical values will be useful as parameters that an optimization algorithm can tweak to find new mappings.

Optimization and Art

Creating art by finding optimal parameters within a constrained space can yield fascinating results. Bosch [1] has many great examples of optimized art which leverages linear programming solvers. But there are also many examples of optimization specific to working with spherical maps. Demaine et al. [2] searched for an optimal foil wrapper shape that can cover a sphere and van Wijk [12] used his myriahedra map projections to find the best way to peel a globe to keeps the land (or oceans) contiguous.

The work that is most relevant to orange peel art optimization is the program by Liu et al. [5] which designs orange peels to a target shape. Their method takes a target planar shape and expands it on the surface of a sphere to be as large as possible without overlaps. It incorporates several user directed interactions to either modify the target image, or to choose a way to modify the current peeled shape. After several iterations of user interaction their software comes up with very satisfying results. The work we show here differs from Liu et al. in a few ways. Our optimization parameters modify a small set of parameters and thus they control only the large scale shape of the peel and not finer details such as individual fingers on a hand. Our method has much less user interaction. Qualitatively, our resulting shapes are a little rougher in the details, but in some ways broadly speaking closer to the target shape. Figure 3 shows a comparison.

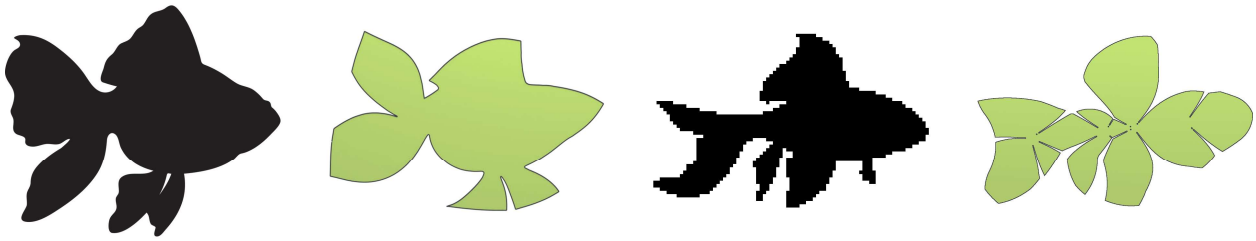

Figure 3: *From left to right: The target shape from Liu et al.; their result; our target shape; our result.*

Our Optimizer

When solving an optimization problem, it is useful to think about several high level aspects of the algorithm. The *objective function* describes our desired result to the algorithm. The *solution space* is a way for the algorithm to explore different possibilities. Many optimization methods need an *initial guess* to start from. And of course the *optimization algorithm* itself informs the solver how to start from the initial guess and navigate through the solution space to find a solution that optimizes the objective function. We discuss each part, one at a time.

Objective Function

To measure how close any given shape S is to a target shape T, we start by overlaying the pixelated versions of S on top of T as shown in Figure 4.

We could set the objective function $f(S, T)$ to the number of pixels in S that are not in T plus the number of pixels in T that are not in S. In data science, this is the false positives plus the false negatives. However, this function would not guide the solver towards the right answer in many cases. For example, if the shapes S and T were completely disjoint, then $f(S, T)$ would be the same no matter how far apart the shapes were. And if we want the objective function to 'point the way' to the correct answer, we would want the error to be lower for closer shapes. To fix this, we have found it useful to weight each erroneous pixel by the distance it is from being correct. We chose to use the Manhattan distance to reduce the amount of computation. Let $d_S(p)$ be the distance from a pixel p to S (or 0 if p is inside S) and similarly with $d_T(p)$ for T. Then our final objective function is $f(S,T) = \sum_{p \in S} d_T(p) + \sum_{p \in T} d_S(p)$.

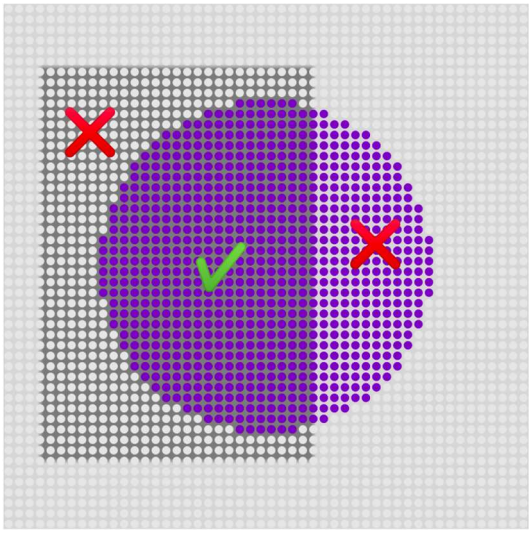

Figure 4: *The shape S (a circle) is pixelated and overlaid on top of target shape T (rectangle). Correct and incorrect pixels are indicated*

Solution Space

As we showed in Figure 2, we specify our orange peel mapping with a skeleton created by a turtle program. We can extract the numerical values of the turtle program, and give them to an optimization program as a set of parameters to tweak. To choose overall topology of the skeletons, we have found through experimentation that tree structures of connected line segments result in the most pleasing orange peel shapes. See Figure 5 for examples. Given such a turtle program, we begin our set of optimization parameters with the angles of each rotation, and the lengths of each line command.

We would like our optimization to be able to adjust the global orientation and offset of our orange peel. So we add an initial *rotate* and *move* command to our turtle program and add their numerical values to the parameter set. Lastly, we found it useful to add a parameter to adjust the global scale of the result. Behind the scenes, global scaling is implemented by multiplying this parameter to the lengths of every *line* command of the turtle on the plane.

Initial Guess

Using a good, feasible solution as an initial guess is important to getting faster, higher quality results. In our case, to determine our initial guess we use a very effective but somewhat naive approach. We start by generating some random skeletons by randomly placing twelve points on the sphere. We use Kruskal's algorithm to find a minimum spanning tree to connect the points with lines and then calculate the turtle program that generates this tree. The skeletons in Figure 5 were generated with this process. After evaluating hundreds of random skeletons using our objective function we simply use the best skeleton found so far as our initial guess.

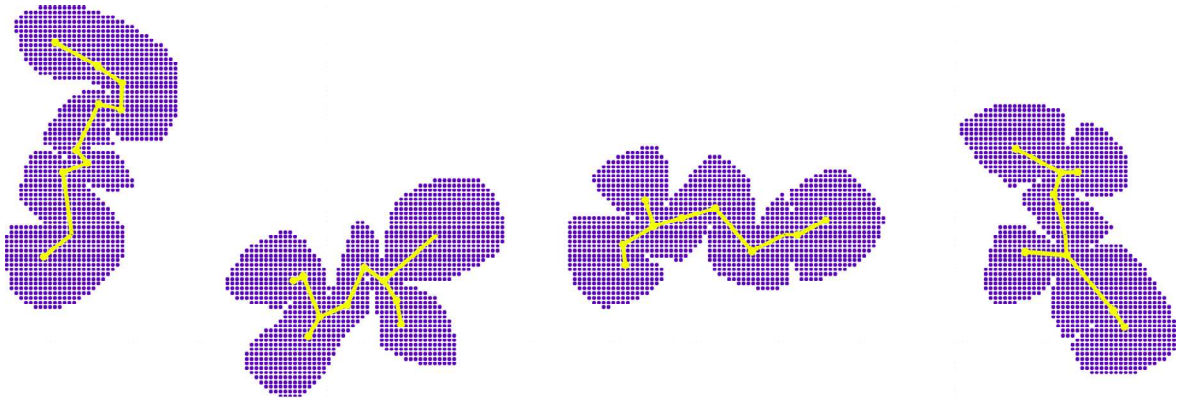

Figure 5: *Random skeletons and their resulting orange peels ready to be optimized.*

Optimization Algorithm

With an objective function, a set of parameters, and an initial guess, our algorithm has everything it needs to search for an optimal solution. Although there are many mathematically sophisticated ways to approach this problem, we use a straightforward gradient descent method. Quite simply we perturb each parameter in turn and if the objective function improves with the new value we keep it. The algorithm continues this way, dynamically updating the step size for each parameter, until no more improvements are to be had.

Our method is not guaranteed, and perhaps not even likely, to find the globally optimum solution. It will, however, find a *local* optima. And with a good enough initial guess, we have found that the resulting orange peel shapes are acceptable for our creative purposes.

Software Implementation

Our program is written in JavaScript and embedded in a standalone web page that can be downloaded and opened on a user's laptop [11]. The readme file there will have further usage instructions, but the basics are: a text box for inputting prewritten turtle programs; controls to select a target silhouette and to optimize the orange peel shape; a low resolution live display to show the current orange peel shape; and the ability to output a high resolution silhouette. Figure 6 shows a use case to see what the algorithm looks like in action. The ability to edit the skeleton with the mouse is a pending feature of this program, as is the ability to map spherical imagery. The runtime of the program takes about seven minutes on an Intel Xeon 4 core 2.80GHz with a graphics card to generate three candidates that a user can select from.

Figure 6: *Top: the target image of a butterfly; the initial guess selected from hundreds of random skeletons; the result after optimization. Bottom: Mapping spherical imagery into a butterfly.*

Results

To exercise our algorithm, and to test the limits of its capabilities, we ran it on a variety of letters. We first tested it on simple letters such as I and L until the bugs were worked out, and then moved on to more complicated letter shapes such as A and G. As expected, the realm of possible shapes turned out to be extremely constrained (e.g., the letter O was particularly difficult, so a filled in circle had to suffice). With trial and error, and taking some liberties, we were able to generate orange peel shapes for the alphabet. Figure 7 shows the resulting font *Orange You Glad*, which is included as a true type font in the supplementary files for this paper.

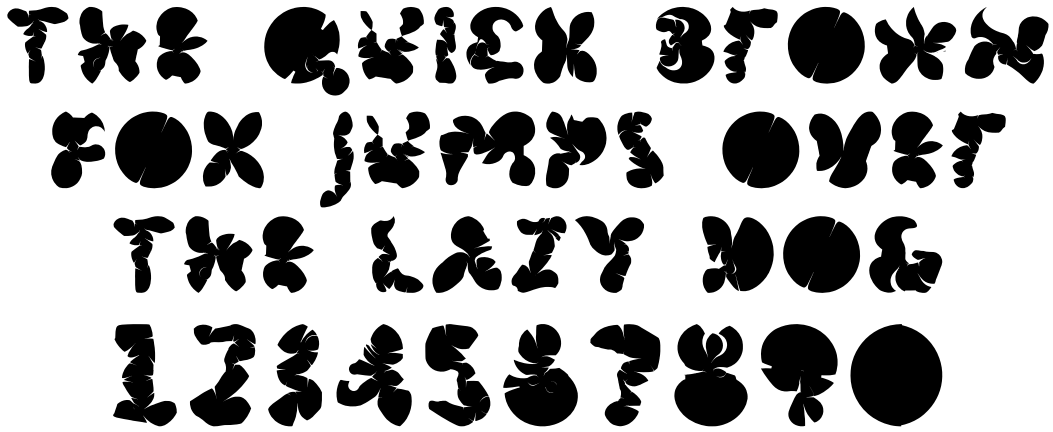

Figure 7: *The orange peel font OrangeYouGlad.ttf*

With this font in hand, we have an opportunity to use the letter designs to peel actual oranges into some words. To transfer the pattern, a printout of each shape was cut out and rolled up into a paper-craft sphere. The locations of the paper seams were transferred to an orange and cut with a knife. Figure 8 shows the results.

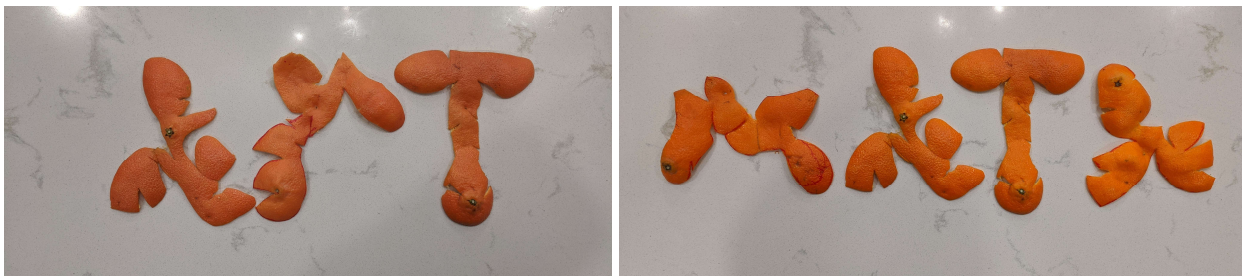

Figure 8: *A physical version of our orange peel font spelling the words* Art *and* Math.

The next project was to use flattened orange peels to mimic the shapes of the major landmasses of the world. In this case spherical images of orange peel textures were mapped to get an orange peel approximation of the Goode homolosine projection (a.k.a., the orange peel map). See Figure 9.

We finish by showing three new designs that finish the series: *Turtle*, *Cat*, and *Goldfish* in Figure 10. Just as *Frog* and *Crow* exhibit the bright colorings of real life species (the blue tree frog and the pied crow), we continue with the natural colorings of a green sea turtle, a calico cat, and a sanke colored goldfish.

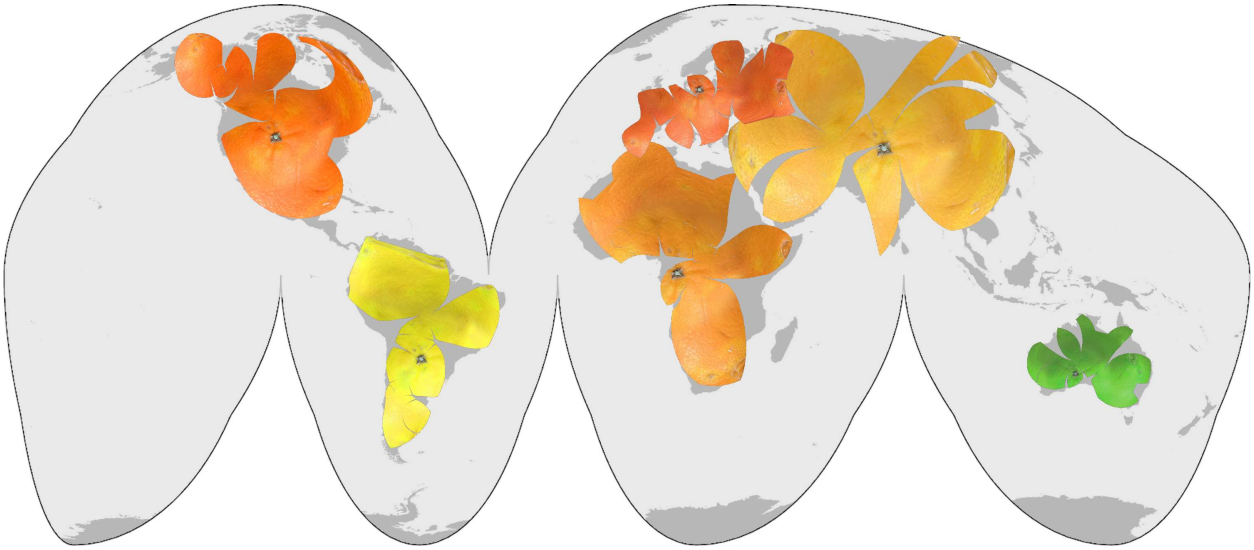

Figure 9: *An orange peel map of citrus peels.*

References

[1] Robert Bosch. *Opt Art: From Mathematical Optimization to Visual Design*. Princeton University Press, 2019.

[2] E. Demaine, M. Demaine, J. Iacono, and S. Langerman "Wrapping Spheres with Flat Paper." *Computational Geometry: Theory and Applications*, vol. 42, no. 8, 2009, pp. 748–757. doi: 10.1016/j.comgeo.2008.10.006.

[3] D. Germán, L. Burchill, A. Duret-Lutz, S. Pérez-Duarte, E. Pérez-Duarte, and J. Sommers. "Flattening the Viewable Sphere." *Computational Aesthetics*, Banff, Canada. Jun. 20–22 2007, pp. 23-28. doi:10.2312/COMPAESTH/COMPAESTH07/023-028.

[4] J. Lee. *Unfold-Lee Jaewon. 2019.* http://leejaewon.net/

[5] H. Liu, X. Zhang, X. Fu, Z. Dong, and L. Liu. "Computational Peeling Art Design." *ACM Transactions on Graphics.* vol. 38, no. 4, 2019, article 64. doi:10.1145/3306346.3323000.

[6] Y. Okada, *Atarashii mikan no mukikata*. Shogakukan, 2010.

[7] S. Pérez-Duarte and D. Swart. "The Mercator Redemption." *Bridges Conference Proceedings*, Enschede, Netherlands. Jul. 27–31, 2013, pp. 217–224. http://archive.bridgesmathart.org/2013/bridges2013-217.html

[8] D. Swart. "Using Turtles and Skeletons to Display the Viewable Sphere." *Bridges Conference Proceedings,* Banff, Canada. Jul. 26–29, 2009, pp. 39–46. http://archive.bridgesmathart.org/2009/bridges2009-39.html

[9] D. Swart. "Papercraft Panoramas." *Bridges Conference Proceedings*, Enschede, Netherlands. Jul. 27–31, 2013, pp. 411–414. http://archive.bridgesmathart.org/2013/bridges2013-411.html

[10] D. Swart and B. Torrence. "Mathematics Meets Photography" In *The Best Writing on Mathematics 2012*, edited by M. Pitici, Princeton University Press, 2013.

[11] D. Swart. *Globemaker*. 2021. https://github.com/dmswart/Personal-Projects/tree/main/globemaker_js

[12] J. van Wijk. "Unfolding the Earth: Myriahedral Projections." *The Cartographic Journal*, vol. 45, no. 1, 2008, pp. 32–42. doi:10.1179/000870408X276594.

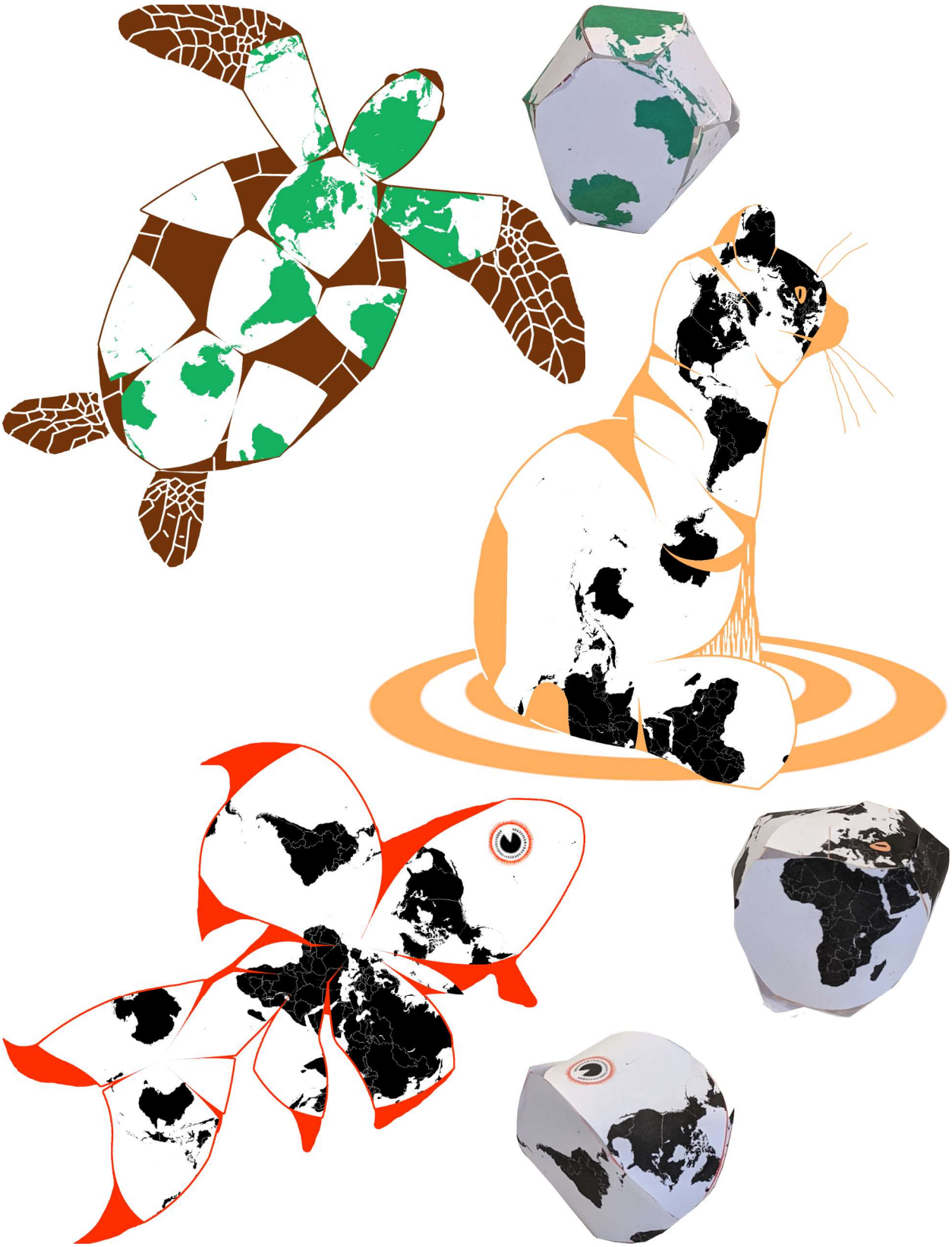

Figure 10: *Designs for three upcoming woodcut prints:* Turtle*,* Cat*, and* Goldfish *and their resulting paper-craft globes.*