IslandPaint: Digital Painting Floating Island Design

Wanwan Li
Department of Computer Science
George Mason University
Fairfax, Virginia, US
wli17@gmu.edu

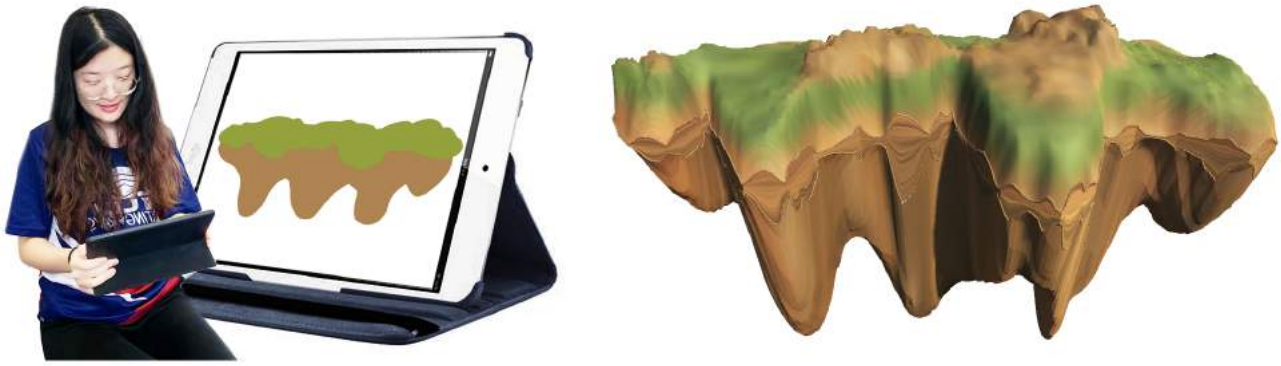


Fig. 1. Demo of IslandPaint: a smart digital painting interface for floating island design. Given the user's arbitrary digital painting of floating island (left), the 3D model of the floating island is automatically generated with our approach (right).

***Abstract*—In this paper, we present a smart digital design interface, IslandPaint: digital painting floating island design. Through IslandPaint, users can design 3D floating islands with simple 2D single-view conceptual digital paintings. As shown in our numerical experiments, after the procedural modeling process proposed by us, 3D floating islands that look like the original 2D paintings can be generated automatically. Given our interface, floating islands design becomes easier for digital art designers, digital multimedia producers, digital movie makers, and digital game authors.**

Keywords*—digital culture; digital art design; digital painting; interactive interface; procedural modeling.

I. INTRODUCTION

With the rapid development of digital multimedia technologies, digital culture becomes a critical part of people's lives. As important ingredients of digital culture, digital arts, digital movies, and digital games are gaining more and more popularity among young people. Computer graphics technology plays an important role in digital art design. With a realistic graphics rendering engine, various types of amazing effects can be visualized realistically on screens, or even on immersive devices. Given these inevitable trends of digital culture's progressing and propagating, computer graphics modeling technologies open people's field of view. The virtual world is beyond the imagination and not beneath the facts. Physics laws in the limited real-world will no longer exist in the entire virtual digital world. The floating island is definitely solid proof of this point of view.

As a direct derivation of the digital culture, floating islands are those islands floating in the sky or space that can never be seen in the natural world. Due to the attractive visual effects and their special existence, floating islands are becoming more and more widely welcomed by digital art designers, digital movie conductors, digital game authors, players, and audiences. Typically, floating islands are the terrain blocks that look like those mountains which are pulled out from the soils and they mostly look like a small part of a terrain that is cut out from a larger part of the terrain.

However, traditional terrain procedural modeling methods are not easily applied for procedural floating island generation. Even though there are some existing works that are aiming at generating floating islands automatically, most of their approaches are based on complex logic flow and have not considered the well-studied terrain features. Also, none of the existing works have considered the interactive user interface for procedural floating island modeling. Therefore, it is challenging to devise an efficient procedural modeling approach to automatically synthesize the floating island from users' conceptual design. In this paper, we present a digital painting-based interface, IslandPaint, to help users design their floating island with conceptual digital paintings. As shown in Figure 1, a demo of our proposed interface is presented: The left subfigure shows a user's original digital painting of a floating island; the right subfigure is the 3D model generated with our approach. Demo video of this example can be accessed at [1]. Contributions of our work include:

- Devising a novel digital painting-based interface for floating island procedural modeling.
- Conducting experiments to demonstrate the results of digital painting-based floating island procedural modeling.
- Discussing the limitation of our approach and proposing future works to extend our interface, to inspire the follow-up works on this research direction.

 ISBN: 978-1-61208-927-0

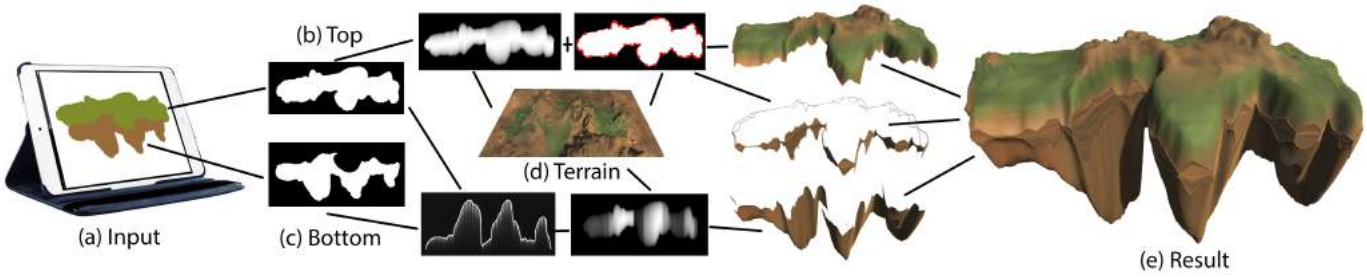


Fig. 2. Overview of our technical approach.

II. Related Works

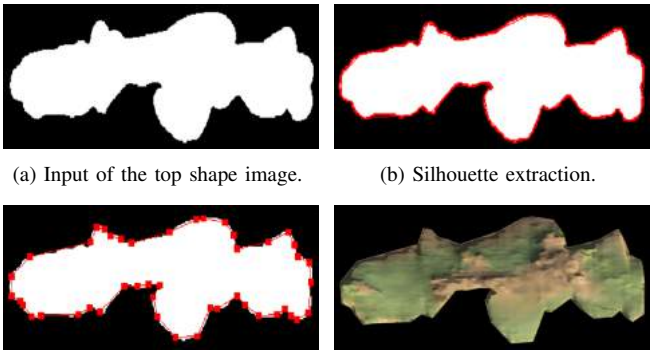
Terrain Procedural Modeling. Lots of terrain procedural modeling technologies have been widely studied in computer graphics research communities. The history of research works on terrain procedural modeling can be dated back to the late 20th centuries. Since 1982, when Fournier et al. [2] propose stochastic models to add fractal details on curves and surfaces, the idea of procedural terrain modeling has born. However, the first work on terrain procedural modeling might have been started even earlier. In 1985, the idea of image texture synthesizer for terrain modeling has been proposed by Perlin et al. [3]. In 1989, the first eroded fractal terrain has been generated by Musgrave et al. [4]. After that, procedural terrain texturing modeling approaches have been widely studied by Ebert et al. [5] in 2003. Since 2007, digital elevation models have been applied to terrain synthesis [6]. Due to advanced technologies in parallel computing, the tasks for generating complex procedural terrain were moved from the CPU to the GPU in 2007 [7]. At the same time, special terrain features, such as spheroidal weathering have been modeled by Beardall et al. [8]. In 2009, with the notion of human-centered computation, interactive user interfaces have been applied onto terrain procedural modeling [9]. At the same time, other complex terrains, such as arches have been successfully modeled by Peytavie et al. [10]. Later, in 2014, procedural generation of 3D canyons has been studied by De et al. [11]. In 2015, parallelity, realism, and controllability have been systematically incorporated in the terrain procedural modeling process. In 2017, volumetric terrain features have been considered in the procedural generation process [12]. Recently, a desertscape terrain generation approach has been proposed by Paris et al. [13] in 2019. At the same time, procedural modeling techniques in riverscapes synthesis have been studied by Peytavie et al. [14]. Most recently, Argudo et al. [15] have systematically simulated the growth of glaciers terrains. Obviously, there is a trend in the research communities that two factors are very important considerations for terrain procedural modeling: one being considering the scientific aspect of the terrain features while another being the interaction and controllability from the users. Therefore, comparing with other existing work on floating island synthesis, our work focuses on these two factors for interactive procedural floating island generation.

Procedural Floating Islands. Designing and generating floating islands or floating continents is an interesting topic for digital movies and games designs. There are lots of works, especially within the digital multimedia design industry, that are focusing on how to generate floating islands smartly. For example, Houdini procedural modeling tutorials [16] have been posed to teach users how to generate floating islands using Houdini [17], a 3D animation software application developed by Toronto-based SideFX. Houdini has been widely adopted as the PRISMS suite of procedural generation software tools. Similarly, 3D Blender [18] tutorials share similar methods [19], [20] and a voxel plugin [21] shows a procedural method to generate voxel-based floating islands. However, all methods are very hard for the users to parameterize an arbitrary floating island from their own will, and the learning curves to master these interfaces are very high for beginners. Therefore, those approaches to generate floating islands lack users' control. On other hand, Houdini or Blender-based procedural modeling approaches have not considered professional procedural modeling approaches [5], [22], [23] studied by researchers. Rather, they all are trying to deform a given manifold surface, such as a sphere, through different sorts of noises, to generate different types of floating terrains. Another related work is proposed by Sandberg et al. [24], but unfortunately, their work is directly deleting the triangles outside the terrain shape and directly copying the bottom mesh from the top mesh, therefore, it results in very poor modeling quality. Although these works can be applied to digital arts or games, the limited control from the users will result in the degradation of the originality and the value of their artworks designs. Different from these existing works, our work focuses more on the user's control over their floating island designs. Also, we model the floating island based on professional terrain procedural modeling approaches.

III. Overview

Figure 2 shows the overview of our approach. Given an image of 2D conceptual digital painting floating island design as input (a), we segment the top image (b) and bottom image (c) according to the color palette, which in this case is grass blue as the top and dark brown as the bottom. Then, we perform separate tasks for the top and bottom images differently. For the top image, we do an image shape distribution analysis

 ISBN: 978-1-61208-927-0



(a) Input of the top shape image. (b) Silhouette extraction.

(c) Polygonization of silhouette. (d) Polygonized terrain mesh.

Fig. 3. Illustration of the terrain shape extraction process.

along the y-axis (the blurred image on top) to get the basemap for the floating island's top heightmap. Simultaneously, we to a contour polygonization as the border shape of the floating island. For the bottom image, we apply a half-distribution analysis of the image shape and then we get a silhouette of the bottom part of the floating island (the white curve on the bottom). Then, we multiply the half-distribution onto the top image so that we get a basemap for the floating island's bottom heightmap (the blurred image on the bottom). Detailed term concepts and mathematical definitions will be expanded and explained in the following section.

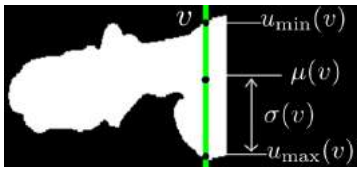
Then, according to the feature map of the generated terrain (d), we synthesize the top mesh of the floating island through the top image, top basemap, and contour polygon (terrain shape). Similarly, we synthesize the bottom mesh of the floating island through the top image, bottom basemap, and contour polygon. In order to avoid any intervals between the top mesh and the bottom mesh, we introduce the middle mesh as a strip that connects the silhouette of the top mesh and the silhouette of the bottom mesh. After the last step of combining these three terrain meshes seamlessly, we get the synthesized floating terrain as the output of our approach (e).

IV. Technical Approach

In this section, we will present the detailed concepts and mathematical definitions for those terms mentioned in the previous section. Note that our approach is proposed based on an important hypothesis that the user's original digital painting design is the abstract conceptual design and there is a loss of depth and texture details. For simplification of the digital image understanding process, we assume the top shape of the original floating island conceptual design is flat.

Terrain Shape Extraction. Given the above assumption, we can extract the terrain shape from the top image in three steps shown in Figure 3. First, as shown in (b), we need to track the silhouette of the white area of the top image using a directional table-based binary image silhouette extraction algorithm [25]. Next, as shown in (c), given these pixels in the extracted silhouette from the top image, we do a line fitting algorithm to approximate the shape of the silhouette as a polygon; We also call this a polygonization process of the top shape. Polygonization is important for reducing the noises on the edge of the synthesized floating island terrain. Then, the last step is to trim the 2D square terrain mesh into polygonized terrain mesh. As shown in (d), we take advantage of the Active Edge Table (AET) polygon filling algorithm [26] and modify it to generate the trimmed terrain mesh with polygon edges [27]. This terrain shape extraction process can be both applied on the top mesh and the bottom mesh synthesis for the procedural floating islands generation.

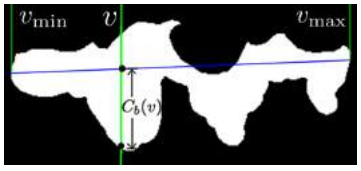
Terrain Basemap Synthesis. The basemap of the terrain is an important concept for the elevation-based terrain procedural modeling techniques. Basemap is typically used for mixing the elevation map of terrain through two different elevation maps, one is the basemap $B(u, v)$ while the other one is the featured heightmap $H_0(u, v)$. Then, the elevation map of the resulting terrain is the $H(u, v) = \alpha(\beta B(u, v) + (1 - \beta)H_0(u, v))$ where $\alpha \in \mathbb{R}$ is the heightmap scale factor and $\beta \in [0, 1]$ is the basemap scale factor. Through this calculation, featured heightmap $H_0(u, v)$ is used for adding features on top of the terrain, while basemap $B(u, v)$ plays an important role in setting the foundation of the terrain. In our work, we consider the basemap to synthesize the terrain heightmap from a given featured heightmap $H_0(u, v)$.

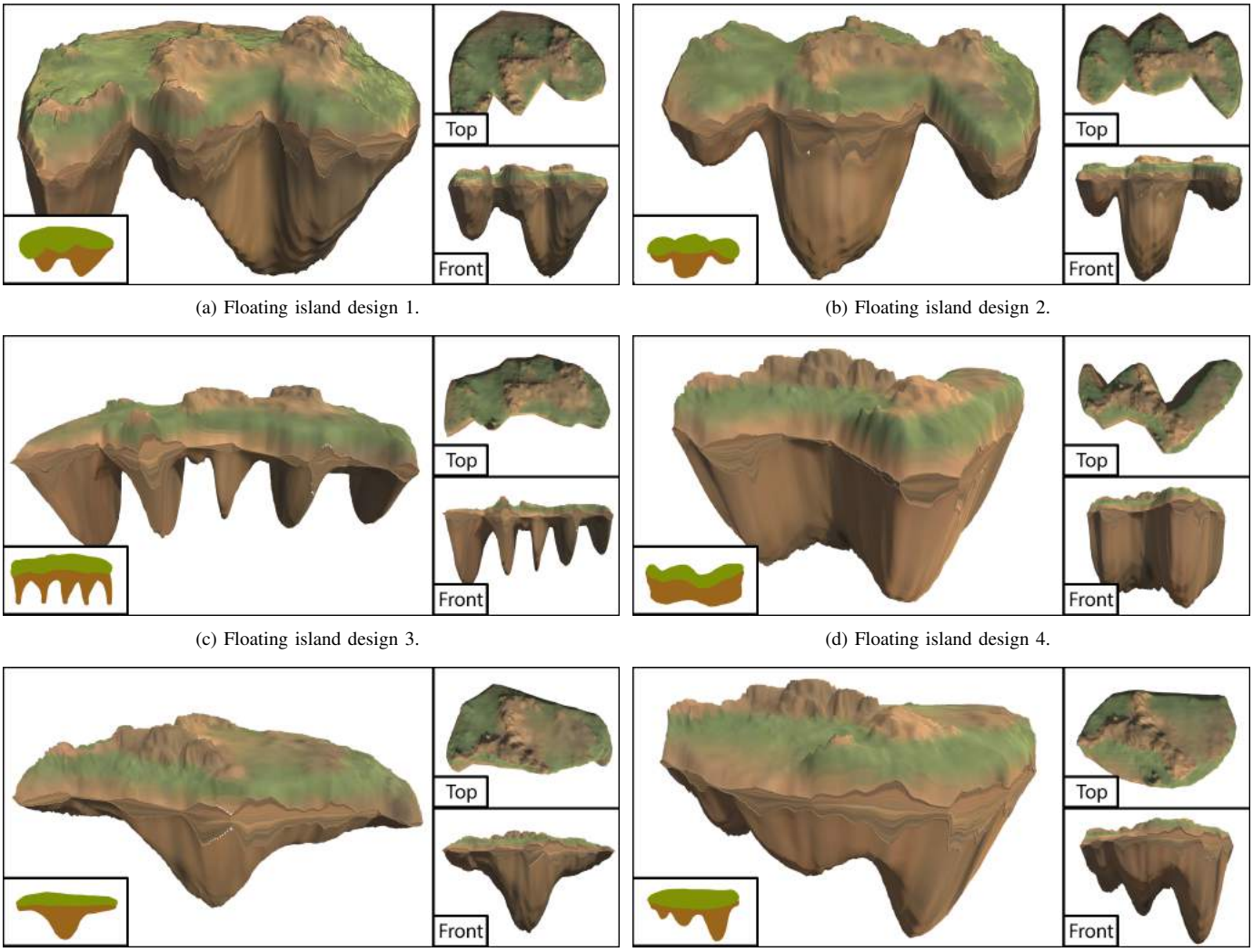


From the observations that floating islands have lower elevations near the edge while higher elevations far away from the edge, therefore, we consider a statistical method to evaluate the basemap B(u, v); We call it image shape distribution analysis. The idea is, given a texture coordinate (texcoord) on a binary image and given a direction, say, v-axis, then the image shape distribution analysis along v-axis will return two functions: shape mean $\mu(v)$ and shape deviation $\sigma(v)$. Let binary image $I(u, v)$ return 1 where texcoord (u, v) is inside the shape; Otherwise, $I(u, v)$ return 0. Then, as shown in the above figure, shape mean $\mu(v)$ is lying on the central curve of the image shape. Mathematically, $\mu(v) = (u_{\min}(v) + u_{\max}(v))/2$ where $u_{\min\,/\,\max}(v) = \min\,/\,\max\{u|I(u, v) = 1\}$. Similarly, $\sigma(v) = (u_{\max}(v) - u_{\min}(v))/2$. Then, we calculate the basemap of terrain $B(u, v)$ as:

$$B(u,v) = \begin{cases} \sin(\cos^{-1}\left(\frac{|u-\mu(v)|}{\sigma(v)}\right)) & \sigma(v) \neq 0 \\ 0 & \sigma(v) = 0 \end{cases} \quad (1)$$

Bottom Shape Analysis. In order to analyze the shape of the bottom part of the floating island, we define another calculation: half-distribution analysis of the image shape. This calculation is similar to the previous distribution analysis process. But, this time, we introduce two texcoords $v_{\min}$ and $v_{\max}$., which is the left most point and right most point on the image shape. Mathematically,





(a) Floating island design 1.

(b) Floating island design 2.

(c) Floating island design 3.

(d) Floating island design 4.

(e) Floating island design 5.

(f) Floating island design 6.

Fig. 4. Experimental results of procedural floating island generations given different user inputs.

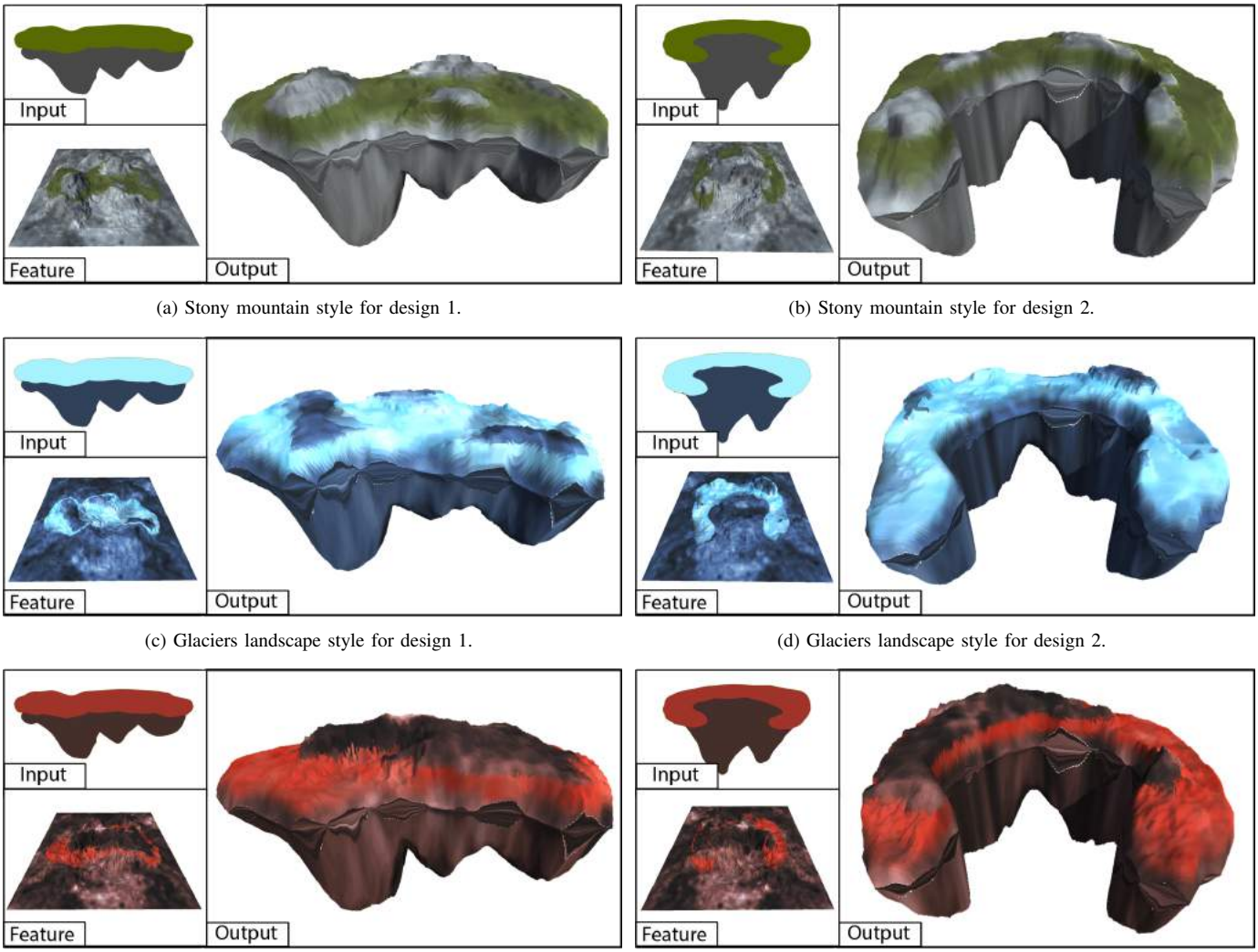
$v_{\min\,/\,\max} = \min\,/\,\max\{v|\exists u \in [0,1] \Rightarrow I(u,v) = 1\}$. Then, as shown in the above figure, the bottom shape curve $c_b(v)$ can be calculated through this formula: $c_b(v) = \mu(v) + \sigma(v) - [(1-t)\mu(v_{\min}) + t\mu(v_{\max})]$, where $t = (v - v_{\min})/(v_{\max} - v_{\min})$. Then, the bottom basemap $B'(u,v) = c_b(v)B(u,v)$, where $B(u,v)$ is the basemap function defined in Equation 1.

Floating Island Generation. After the top basemap and bottom basemap are calculated, the floating island is ready to be assembled from those elevation maps. The top heightmap of the floating island is $H_{\text{top}}(u,v) = \alpha_{\text{top}}(\beta_{\text{top}}B(u,v) + (1 - \beta_{\text{top}})H_0(u,v))$. As the bottom heightmap of the floating island is beneath the sea level, therefore $\alpha_{\text{bottom}} < 0$. Then, we have the bottom heightmap of the floating island as: $H_{\text{bottom}}(u,v) = \alpha_{\text{bottom}}(\beta_{\text{bottom}}B'(u,v) + (1-\beta_{\text{bottom}})H_0(u,v))$. After the final generation of the top mesh and buttom mesh using these heightmap equations, we add another mesh called middle mesh to make up the intervals between these two meshes. Then, the final floating island is generated with our approach.

V. Experimental Results

In order to validate the effectiveness of our approach, we have conducted a series of computational experiments. As shown in Figure 4, we have collected six different digital paintings of floating island design. Given these designs, we run our algorithms to automatically generate the floating islands that are resembling the original input designs. We have implemented our algorithms on Unity 3D with the 2019 version. The hardware configurations contain Intel Core i5 CPU, 32GB DDR4 RAM, and NVIDIA GeForce GTX 1650 4GB GDDR6 Graphics Card. Figure 4 shows the results of the procedural floating island generations with the following settings. For the top terrain mesh, the settings are: heightmap scale $\alpha_{\text{top}} = 0.2$; basemap scale $\beta_{\text{top}} = 0.2$. For the bottom terrain mesh, the settings are: heightmap scale $\alpha_{\text{bottom}} = 0.8$; basemap scale $\beta_{\text{bottom}} = -2.5$. The terrain feature heightmap $H_0(u,v)$ is generated with the standard canyon filters. Subfigures are showing the top view and front view, respectively, for each synthesized 3D floating island terrain.

 ISBN: 978-1-61208-927-0



(a) Stony mountain style for design 1.

(b) Stony mountain style for design 2.

(c) Glaciers landscape style for design 1.

(d) Glaciers landscape style for design 2.

(e) Volcanic landscape style for design 1.

(f) Volcanic landscape style for design 2.

Fig. 5. Experimental results of changing terrain features and textures. In this figure, we present the visual effects when applying different types of terrain features on the same digital painting input. The first column shows the synthesized floating islands on the first digital painting input with different terrain features. The second column shows the synthesized floating islands on the second digital painting input with different terrain features. The first row shows the synthesized floating islands with stony mountain-style features for the first design and the second design, respectively. The second row and the third row show the synthesized floating islands with the glaciers landscape style and the volcanic landscape style, respectively.

As we can see from Figure 4, the generated floating islands are matching well with the user's conceptual digital painting designs. For example, in subfigure (b), we can see three spherical blocks specified in the user's original digital painting design as plotted in the left-bottom corner where the blocks on two sides are smaller than the center block. The same effects appear in the result. Also, in order to consider the foreshortening effects on the digital painting concept design from an orthogonal view, we provide an automatic scaling step to stretch the top image (green part) so that the 3D results are zoomed-in along the z-axis of depth direction. By default, the scaling factor is set to 1.5 and this effect is obvious in subfigure (d). Users can also manually set up this scaling factor to adjust the synthesized 3D outputs to satisfy their expectations.

Changing Terrain Features and Textures. Besides testing our proposed interface with different user inputs, we have tested the robustness of our approach on different types of terrain features and terrain textures. As shown in Figure 5, with the same terrain settings as claimed before, different terrain features are added onto the generated floating islands. In this experiment, we take two different digital paintings from users as inputs. For each input, we applied three different types of terrain features and textures. These are: stony mountain style textures, glaciers landscape style textures, and volcanic landscape style textures, respectively. As shown in the results, our approach can not only generate realistic floating islands according to users' different digital paint designs, but also can be able to add different types of terrain features.

 ISBN: 978-1-61208-927-0

VI. Conclusions

In this paper, we present IslandPaint, a smart user interface for digital painting-driven floating island design. In order to let users efficiently design the 3D floating islands with simple 2D single-view conceptual digital paintings, we proposed a novel approach to automatically extract the 3D information hidden in the conceptual designs. We first propose a hypothesis that the users' paintings are focused on flat floating islands. Then, we segmented the top shape and bottom shape using an image segmentation algorithm. In the next steps, we extract the polygon geometry from the top image and extract the height information from the bottom image. Therefore, we can reconstruct the floating islands whose top views are matching with the top image correctly while the front views are matching with the bottom image correctly. Then, after the procedural modeling process proposed by us, 3D floating islands that look like the original 2D paintings will be automatically generated. As shown in the experimental results, we validated our approach through different user's digital painting designs and the results look promising. At the same time, we tested our interface on different types of terrain features. Both results show that our approach can be compatible and extended with existing terrain procedural modeling technologies very well.

However, there still are some limitations in our work. First, our approach is based on the hypothesis that the user's digital paintings are merely referred to as those floating islands that have flat top surfaces. As a matter of fact, there are lots of floating island design works that are referring to the bumping terrains. Therefore, our approach will not be able to work correctly on these scenarios. In order to solve this, it will rely on proposing an optimization framework to extract the 3D information from the 2D paintings by minimizing the costs functions that are evaluating how well the reconstructed 3D information matches with the perceived 2D information. This is a challenging topic and is worthy to explore as future work. On the other hand, our interface does not allow users to add too many details on the terrain surface. This limited the freedom of degree on user's artistic creations. However, by adding more details of the design, such as wrinkles, the degree of the user's control over the floating island design process will improve significantly. This is another challenging topic to try as the follow-up of this research work.

According to the experimental demonstration of our proposed interface presented in this paper, we believe that, using the interface of IslandPaint proposed by us, floating islands design will become easier for digital art designers, digital multimedia producers, digital movie makers, and digital game authors in the near future. Also, we believe that our work opens an interesting research topic on interactive procedural floating island design and will attract more researchers to further explore academic studies along this direction and follow up with the technical approaches presented in this paper.

References

[1] W. Li, "Islandpaint: Digital painting floating island design," https://youtu.be/YXgmF89UIvY, Nov 2021.

[2] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Communications of the ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[3] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[4] F. K. Musgrave, C. E. Kolb, and R. S. Mace, "The synthesis and rendering of eroded fractal terrains," *ACM Siggraph Computer Graphics*, vol. 23, no. 3, pp. 41–50, 1989.

[5] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[6] H. Zhou, J. Sun, G. Turk, and J. M. Rehg, "Terrain synthesis from digital elevation models," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 4, pp. 834–848, 2007.

[7] R. Geiss, "Generating complex procedural terrains using the gpu," *GPU gems*, vol. 3, pp. 7–37, 2007.

[8] M. Beardall, M. Farley, D. Ouderkirk, J. Smith, M. Jones, and P. K. Egbert, "Goblins by spheroidalweathering." in *NPH*, 2007, pp. 7–14.

[9] G. J. De Carpentier and R. Bidarra, "Interactive gpu-based procedural heightfield brushes," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009, pp. 55–62.

[10] A. Peytavie, E. Galin, J. Grosjean, and S. Mérillou, "Arches: a framework for modeling complex terrains," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 457–467.

[11] D. M. De Carli, C. T. Pozzer, F. Bevilacqua, and V. Schetinger, "Procedural generation of 3d canyons," in *2014 27th SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE, 2014, pp. 103–110.

[12] M. Becher, M. Krone, G. Reina, and T. Ertl, "Feature-based volumetric terrain generation," in *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2017, pp. 1–9.

[13] A. Paris, A. Peytavie, E. Guérin, O. Argudo, and E. Galin, "Desertscape simulation," in *Computer Graphics Forum*, vol. 38, no. 7. Wiley Online Library, 2019, pp. 47–55.

[14] A. Peytavie, T. Dupont, E. Guérin, Y. Cortial, B. Benes, J. Gain, and E. Galin, "Procedural riverscapes," in *Computer Graphics Forum*, vol. 38, no. 7. Wiley Online Library, 2019, pp. 35–46.

[15] O. Argudo, E. Galin, A. Peytavie, A. Paris, and E. Guérin, "Simulation, modeling and authoring of glaciers," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–14, 2020.

[16] H. Beyer, "Procedural floating islands - houdini," https://www.youtube.com/watch?v=jzdACcJSl9c, Aug 2019.

[17] E. Goffredo, "A tool for procedural destruction in houdini shattering+ dynamics," *Masters. Bournemouth University*, 2010.

[18] L. Flavell, *Beginning blender: open source 3d modeling, animation, and game design*. Apress, 2011.

[19] Imphenzia, "Let's make a floating island in 10 minutes in blender," https://www.youtube.com/watch?v=Wh7QD-3ZgLw, Jan 2020.

[20] S. F. Humanity, "Blender 2.9 - procedural floating islands with sorcar addon (sfh5 wip)," https://www.youtube.com/watch?v=EeOt_Ubt6E4, Jul 2020.

[21] J. Kelly, "Floating islands," https://wiki.voxelplugin.com/Floating_Islands, October 2020.

[22] R. M. Smelik, K. J. De Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen, "A survey of procedural methods for terrain modelling," in *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, vol. 2009, 2009, pp. 25–34.

[23] E. Galin, E. Guérin, A. Peytavie, G. Cordonnier, M.-P. Cani, B. Benes, and J. Gain, "A review of digital terrain modeling," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley-Blackwell Publishing Ltd., 2019, pp. 553–577.

[24] R. Sandberg, "Generation of floating islands using height maps," 2013.

[25] Y.-L. Lin and M.-J. J. Wang, "Automated body feature extraction from 2d images," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2585–2591, 2011.

[26] M. R. Dunlavey, "Efficient polygon-filling algorithms for raster displays," *ACM Transactions on Graphics (Tog)*, vol. 2, no. 4, pp. 264–273, 1983.

[27] L. A. Piegl and A. M. Richard, "Tessellating trimmed nurbs surfaces," *Computer-Aided Design*, vol. 27, no. 1, pp. 16–26, 1995.

 ISBN: 978-1-61208-927-0