Visual Informatics 6 (2022) 23–34

Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

Example-based large-scale marine scene authoring using Wang Cubes

Siyuan Zhu [a], Xinjie Wang [a,*], Ming Wang [b], Yucheng Wang [b], Zhiqiang Wei [a], Bo Yin [a], Xiaogang Jin [c]

[a] Ocean University of China, Shandong, 266100, China
[b] Pilot National Laboratory for Marine Science and Technology (Qingdao), Shandong, 266237, China
[c] Zhejiang University, Zhejiang, 310058, China

ARTICLE INFO

Article history:
Received 4 January 2022
Received in revised form 12 May 2022
Accepted 21 May 2022
Available online 2 June 2022

Keywords:
Wang Cubes
Marine scene authoring
Virtual environments

ABSTRACT

Virtual marine scene authoring plays an important role in generating large-scale 3D scenes and it has a wide range of applications in computer animation and simulation. Existing marine scene authoring methods either produce periodic patterns or generate unnatural group distributions when tiling marine entities such as schools of fish and groups of reefs. To this end, we propose a new large-scale marine scene authoring method based on real examples in order to create more natural and realistic results. Our method first extracts the distribution of multiple marine entities from real images to create Octahedral Blocks, and then we use a modified Wang Cubes algorithm to quickly tile the 3D marine scene. As a result, our method is able to generate aperiodic tiling results with diverse distributions of density and orientation of entities. We validate the effectiveness of our method through intensive comparative experiments. User study results show that our method can generate satisfactory results which are in accord with human preferences.

1. Introduction

Natural scene simulation has been receiving considerable attention in computer animation and computer simulation. As marine scenes are commonly found in natural environments, authoring and generating marine scenes are valuable in the fields of 3D visualization of marine data, film production, and science education, etc. In order to create realistic and immersive virtual marine environments, it is necessary to populate an environment with marine entities such as fish, marine submerged plants, and reefs while ensuring that the distribution of these entities is as close as possible to real environments. However, research on marine scenes mainly focuses on physical simulation fields such as sea surface shading, wave motion, refraction, and caustics. There is no prior work on large-scale marine entity populating methods. In practical applications, we expect users to quickly author diverse marine scenes with naturally distributed marine entities by only adjusting a few parameters in a simple way.

Our goal is to provide an interactive authoring tool that can quickly generate a scene with large-scale marine entities based on the example marine entities and virtual models provided by the user. The generated scene should have the following design objectives:

- *Behavioral diversity.* The behavior of creature groups (e.g., fish schools) should have diversification such as alignment, aggregation, hide, and escape. Moreover, these behaviors should not have obvious periodic patterns.
- *Species (objects) diversity.* The scene should contain a variety of creature groups, and they should have group interactions. In addition, the scene should contain non-living marine entities, such as marine submerged plants and reefs, to make the marine scene more vivid.
- *Performance.* The scene generation should be fast enough to facilitate users to create and edit scenes quickly.

In this paper, we try to leverage group (crowd) simulation techniques to accomplish this task. Methods to generate large-scale groups mainly consists of two categories: continuous models and discrete models. However, these group simulation models usually do not satisfy all three objectives described above at the same time, and therefore they do not meet our expectations for marine scene authoring.

Continuous models are macro-level simulations that take the group behavior as the study object. These models usually use field potential or fluid dynamics to control the behaviors of the groups (Treuille et al., 2006; Narain et al., 2009). However, continuous models usually focus on the overall behavior and ignore

* Corresponding author.
E-mail addresses: zhusiyuan@stu.ouc.edu.cn (S. Zhu), wangxinjie@ouc.edu.cn (X. Wang), mwang@qnlm.ac (M. Wang), ycwang@qnlm.ac (Y. Wang), weizhiqiang@ouc.edu.cn (Z. Wei), ybfirst@ouc.edu.cn (B. Yin), jin@zju.edu.cn (X. Jin).

https://doi.org/10.1016/j.visinf.2022.05.004

the diversity of individual species. In addition, continuous models are usually designed to guide groups in path planning. In our scenario, marine scenes contain a variety of independent groups with no specific path, so these continuous models are not suitable for the authoring of marine scenes.

Discrete models are micro-level simulations that take individuals as the study object. Each individual is treated as an independent agent, and these agents are influenced by the behavior of surrounding agents and also the environment, making the agents interactive, and thus revealing group behaviors (Hartman and Benes, 2006; Silva et al., 2010; Helbing et al., 2005; Snape et al., 2011). However, existing discrete models for large-scale group generation often result in groups exhibiting the same behavior or having the same movement path between groups, i.e., periodic patterns. As a result, they may generate unsatisfactory results when they are used to author marine scenes.

In addition, another important class of research in 3D scene simulation is indoor authoring (Zhang et al., 2019), which focuses on the automatic arrangement of furniture objects. With the gradual enrichment of data sets, techniques for indoor authoring have shifted to data-driven approaches based on which they learn a prior knowledge of object distribution through images, and thus ensure the quality of layouts (Zhang et al., 2021; Fisher et al., 2012; Xu et al., 2013). Compared to indoor scene authoring, natural scene simulation is richer in species and more complex in distribution.

Addressing the above design objectives may bring the following challenges. First, it is not enough to rely on a number of predefined rules or stochastic algorithms to generate aperiodic patterns of diverse group behaviors and group species. Second, due to the large and diverse marine groups, authoring individuals independently in 3D space can be very computationally intensive and does not meet the need for fast editing.

We tackle these challenges by developing an example-based method for authoring large-scale marine scenes efficiently. Our method can be considered as a meso-level model which is between macro-level and micro-level. In order to achieve realistic marine group behaviors, inspired by data-driven approaches, we extract the distribution examples of marine entities from authentic marine images. Then, these examples are crafted into a set of local feature Octahedral Blocks for aperiodic tiling. Each Octahedral Block contains collective behavior information of one or more sets of marine entity groups, such as distribution, orientation, and species type. Next, these Octahedral Blocks are cut and assembled by following Wang Cubes (Wang, 1961) generation rules to produce a set of Marine Cubes. Finally, the marine scene is quickly tiled in 3D space using the Marine Cubes and user-defined parameters.

In summary, our paper makes the following main contributions:

- We propose a first-of-its-kind example-based method for authoring marine creatures (objects) and marine group behaviors effectively. Our approach leverages real examples to create final large-scale marine scenes. Both the experimental results and user study results show that our method is able to generate marine scenes naturally and efficiently. To the best of our knowledge, it is the first approach to author large-scale realistic marine sciences in real time using exemplars.
- Unlike previous techniques, our approach allows authoring both 2D and 3D mixed groups (crowds) simultaneously.
- We propose a point-set based spatial density adjustment method for interactive design. It can dynamically adjust the creature group distribution during the tiling process according to the characteristics of different species, such as the survival depth and survival probability, which significantly improves the naturalness of the generated results.

To evaluate the effectiveness of our method, we have generated marine scenes for a variety of scales, where scenes with hundreds of thousands of individuals took only a few seconds to generate. In addition, by using two metrics proposed in Yang et al. (2020), we also quantitatively compare the behavior of our method with the Boids method (Reynolds, 1987). The comparison results show that our method has competitive results with the Boids method in generating fish schools. Finally, we conducted a user study to show that our results meet natural and realistic group authoring results and match human preferences.

2. Related work

This work aims to populate the environment with large-scale marine groups efficiently. Our task involves group (crowd) simulation and scene synthesis methods. As addressed in the previous section, group (crowd) simulation can be classified into three categories due to different spatial resolutions: macro-level, meso-level, and micro-level. As there have been many previous works on group (crowd) simulation, we only discuss the most relevant previous work for brevity, including data-driven crowd simulation and 3D group simulation. As for scene synthesis techniques, we mainly discuss Wang Cubes and its related synthesis techniques.

2.1. Data-driven crowd simulation

Data-driven techniques can be used to extract behavioral data from the real world for crowd simulation. These techniques can effectively enhance the realism and accuracy of simulation results. In recent years, data-driven crowd simulation has achieved pleasing results in crowd simulation and traffic scenario simulation. In crowd simulation, Lee et al. (2007) proposed a simulation method that uses 2D pedestrian trajectory data extracted from natural videos. During the simulation, agents learn these trajectory data and decide their behaviors based on the environment and neighboring agents' behaviors. Their experimental results show that the simulation results are similar to the real video. Ju et al. (2010) used real crowd videos to sample data on crowd trajectories and then used a rule-based approach to simulate an arbitrary number of people. Kim et al. (2016) proposed a crowd simulation method combining data-driven techniques with the adaptation and interactivity of synthetic multi-agent techniques. The method generates interactive virtual pedestrians after capturing pedestrian motion features from real-world videos. Karamouzas et al. (2017) proposed an energy function-based crowd velocity optimization method. The method can generate stable and high-quality simulations of crowds of different densities with guaranteed collision-free performance. Ren et al. (2021) proposed Heter-Sim, and it combines data-driven and physics-based simulation methods to achieve heterogeneous agents with different combinations of agents for different dynamics of agent objects. The system was successfully applied to the simulation of crowd and traffic scenarios.

In the traffic simulation, Chao et al. (2013) extracted the motion feature information of vehicles in the video and applied it to large-scale vehicle animation. The method can follow the surrounding changes to adjust the driving behavior and has a strong sense of realism. Subsequently, Chao et al. (2017) proposed a texture-synthesis and data-driven approach for traffic flow simulation by inputting a limited number of real-world vehicle trajectories, viewing the traffic flow as a two-dimensional texture, and generating a large number of vehicle trajectories using texture synthesis. Bi et al. (2016) proposed a data-driven traffic lane change model. They extracted features of the target vehicle and surrounding vehicles from real-world traffic flow and

input them to a back-propagation network for building a vehicle trajectory lane change decision model. The model makes lane changes in traffic flow close to the real world and improves the realism of traffic flow. Li et al. (2017) reconstructed the traffic from GPS data and used metamodel-based simulation to optimize the data deficient areas.

However, these data-driven approaches are designed for crowds or traffic flows, not for animals. Moreover, they usually focus on two-dimensional space. They cannot be directly extended to 3D marine group simulations. In the following, we will introduce 3D group simulation.

2.2. 3D group simulation

The main group simulations in 3D space are the flock of birds, the swarm of insects, and schools of fish.

In 1987, Reynolds (1987) proposed the Boids method, a rule-based flock simulation algorithm, which uses three principles of collision avoidance, center proximity, and velocity matching to constrain the flock. With further research (Hartman and Benes, 2006; Alaliyat et al., 2014; Paranjape et al., 2018; Iizuka et al., 2018; Inomata and Takami, 2020; Xueying et al., 2017), the Boids method was gradually improved and used until now. Meanwhile, Vicsek et al. (1995) use the principles of statistical mechanics to calculate the forces and directions of individuals in a group. Since the Vicsek model can simulate complex systems with simple principles, researchers have applied it to the simulation of bird flocks other groups (Ginelli, 2016; Xueying et al., 2017; Chen et al., 2021).

Besides the simulation of bird flocks, insect swarm simulation is also a trendy research area in group simulation. For example, Wang et al. (2014) proposed a field-based insect simulation method. Ren et al. (2016) proposed a data-driven formulation based on pre-recorded insect trajectories for collective insect behavior, including aggregation, migration, phase change, and escape. Chen et al. (2019) proposed an external force model based on a trade-off mechanic that allows insects to move into specified shapes naturally. Xiang et al. (2020) proposed the FASTSWARM framework for high-fidelity simulation of common insect behaviors. This method can be used to predict the missing insects' trajectories in the captured data set, and they also provide a user-controlled interface for editing simulating results. However, these methods focus on one or several groups, while many organism groups usually exist simultaneously in a marine scene. Therefore, a direct use of these methods may produce identical or similar behaviors in marine groups that do not exhibit natural behavioral features.

In fish school simulation, the artificial life technique is the most common method, i.e., artificial fish. Xiaoyuan's fish (Tu and Terzopoulos, 1994) is the first generation of the artificial fish model. Artificial fish are generated using a cognitive-based approach and are capable of exhibiting intelligent behaviors, including predation, escape, and aggregation. However, since artificial fish algorithms are usually computed individually on each fish, they do not perform well when conducting large-scale fish school simulations (Zhou et al., 2015). Therefore, the artificial fish algorithm is not applicable when authoring marine scenes interactively.

2.3. Wang Tiles and Wang Cubes

Wang Tiles is a synthetic algorithm proposed by Hao Wang in 1961 (Wang, 1961), mainly for solving the tessellation problem of planes. In 1966, Robert proved the tessellation planes generated by Wang Tiles are aperiodic (Berger, 1966). Nowadays, Wang Tiles are widely used in the work of creating pattern textures (Stam, 1997; Cohen et al., 2003), terrain textures (Derouet-Jourdan et al., 2016), and group scenes (Shen et al., 2014). These works make use of limited data to obtain aperiodic data results. Specifically, in the scene authoring area, Shen et al. (2014) proposed a fast sample-based crowd authoring method using Wang Tiles for fast crowd synthesis. Their method was able to create natural crowds with rich local behavior. However, the method is still in 2D space and does not apply to author 3D marine scenes.

Wang Cubes (Culik and Kari, 1996) is a 3D extension of Wang Tiles, which can be used to create aperiodic tessellation in 3D space. Based on this method, Sibley et al. (2004) proposed a Wang Cubes-based video synthesis method that uses sampled video or Poisson distributed points Wang Cubes to perform video synthesis or geometry placement in a short time. The method has achieved good results for focal dispersion creation of shallow pool scenes and asteroid belt creation. However, their method requires video stream as inputs, but the lack of marine video resources makes the method not applicable to author marine scenes. In addition, Lu et al. (2007) developed a Wang Cubes-based interactive body illustration system to author human organs using data sets such as feet and abdomen. Their method has the advantages of creating local details of the human body, such as fingerprints of the feet and subcutaneous blood vessels, but these advantages are not necessary for generating marine scenes. It is worth mentioning that our method is designed for marine group authoring, not only using Wang Cubes to create aperiodic groups efficiently but also proposing a point-set based spatial density adjustment method for resolving artifacts caused by overstocking of marine groups.

3. Method

In this section, we will describe our example-based method for authoring marine scenes. The overview of our method is shown in Fig. 1. First, we create a library of marine entities, including species such as marine creatures (take fish schools as an illustration in our experiments), submarine plants, reefs, etc. Then, we fill the marine scene quickly by using the species in this library. Specifically, the primary process is described as follows:

- *Octahedral Blocks Creation.* We extract the distributions of marine entities from real images (denote as *examples*) and create the *Octahedral Blocks* by using these examples.
- *Marine Cube Set Synthesis.* Several specified Octahedral Blocks are combined to form a *Marine Cube.* All Marine Cubes form a Marine Cube set.
- *Scene Tiling.* We further use the Marine Cube set to tile the marine scene in the order of X, Y, and Z axis, respectively. The coordinates of the entities in all selected Marine Cubes are updated.
- *Species and Orientation Randomization.* Every marine entity is assigned a species number, and a 3D vector is also randomly generated as its orientation.
- *Density adjustment.* To prevent artifacts caused by overstocking of marine groups, we use a point-set based approach to dynamically adjust the density of the generated scene.

3.1. Octahedral Blocks creation

The creation of Octahedral Blocks requires two sub-steps.

First, we extract the real distribution of marine entities as input examples by taking a data set of real marine images as references. Meanwhile, we also set a corresponding entity information table for each marine entity in the examples, including their coordinates, sizes, species types, group numbers, survival depths (denoted as d), and survival probabilities (denoted as s).

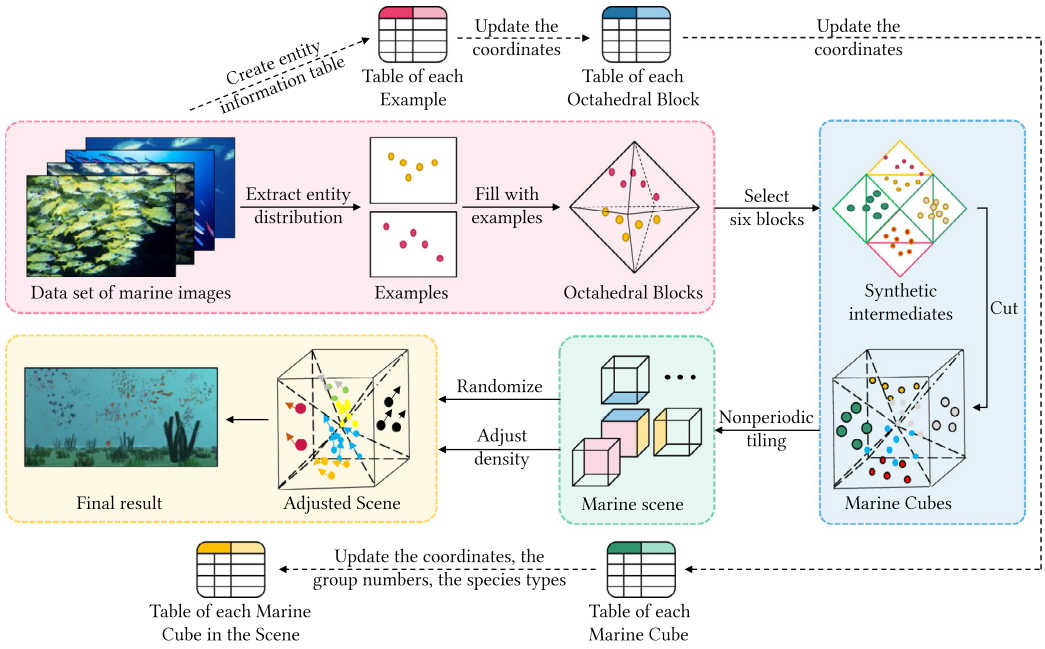


Fig. 1. The overview of our method. Firstly, we create a real data set of the marine images. Secondly, we extract the distribution information of marine entities in these data sets as examples and create an entity information table for each example. Next, we fill these examples into Octahedral Blocks and update the coordinates in the entity information table of the examples as the Octahedral Blocks' table. Then, we select Octahedral Blocks to synthesize Marine Cubes and update the coordinates in the entity information table of the Octahedral Blocks as the Marine Cubes' table. Following, specific Marine Cubes are taken from Marine Cubes to tile the marine scene. In the meantime, the coordinates, the group numbers, and the species type in the entity information table are also updated for the scene. Finally, we randomize the species and orientation of the entities and adjust the density of the scene.

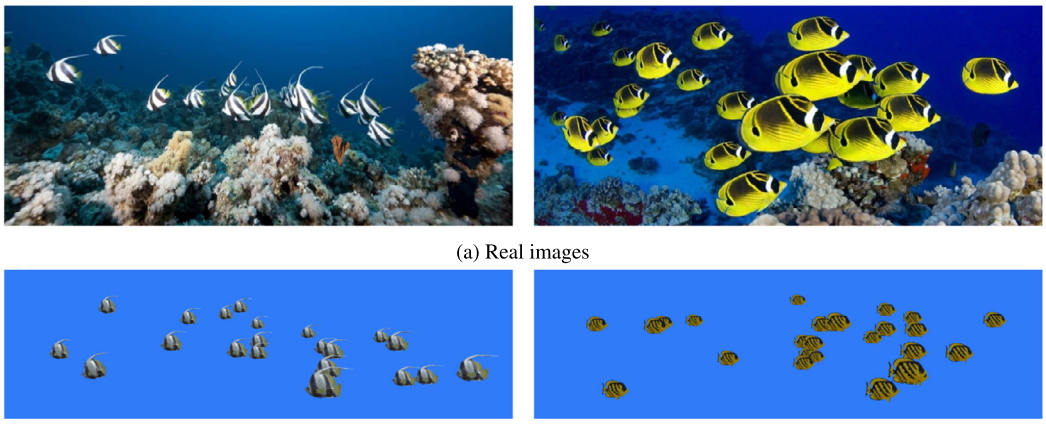


Fig. 2. Extraction of distribution information from real images. (a) Real images of Pennant coralfish and Chaetodon lunula. (b) Corresponding examples of these fish.

During the extraction, the behavior and distribution of entities in each group (for example, a school of fish) should match the real images. Specifically, we determine how many species of fish are in the image and select the corresponding number of models, then we estimate the size of the fish population in the image to determine the placement area, and finally, we place the corresponding models in this area according to the distribution of fish in the image. An illustration of extraction is shown in Fig. 2. Fig. 2(a) shows the real image of the Pennant coralfish and the Raccoon butterflyfish. The corresponding examples are shown in Fig. 2(b), respectively.

Next, we fill the examples into a series of octahedron-shaped blocks (non-regular, see Fig. 3) called *Octahedral Blocks*. The size of the Octahedral Block can be automatically generated by the system based on the scene size or specified by the user. According to the different orientation of the symmetry axis of an Octahedral Block to the X, Y, and Z axis of the world coordinate system, there are three types of Octahedral Blocks: left (right), up (down) and front (back), as shown in Fig. 3. Each Octahedral Block can be filled with an arbitrary number of examples. But it is preferable to fill it with a number that matches the density of the natural distribution of marine entities. Octahedral Blocks can also be filled empirically; that is, the user provides input parameters such as the number of schools and the distance range between groups of the specified marine entities to fill the blocks. Since each layer of the marine scene contains different species, we also need to classify the submarine and non-submarine Octahedral Block set, making it possible to select the correct Octahedral Blocks during the tiling process. To do so, we fill the up (down), front (back), left (right) Octahedral Blocks with entities to obtain the non-submarine Octahedral Block sets, named as UBlocks, FBlocks, and LBlocks. We fill the front(back), left(right) Octahedral Blocks with reefs, seaweeds, and Near-bottom entities to obtain the submarine Block sets, named as GFBlocks and GLBlocks. In addition, it is

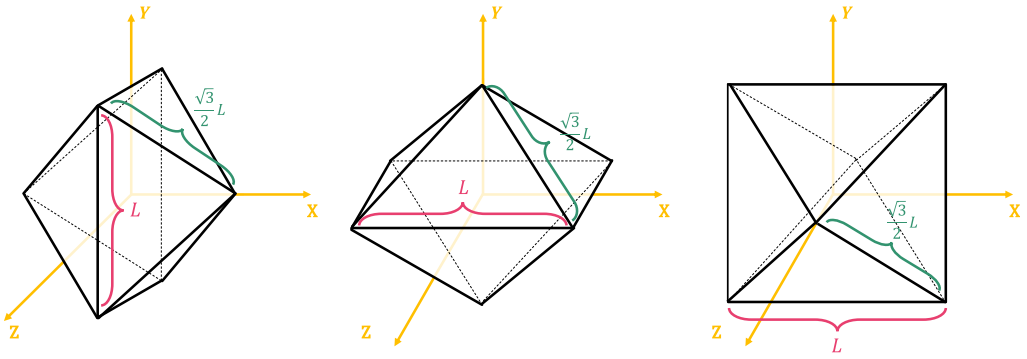


Fig. 3. Three types of Octahedral Blocks with respect to the X, Y and Z axis: left (right), up (down), and front (back).

necessary to conduct an intersection test on the marine entities in each Octahedral Block to ensure that entities are all within the block. The intersection test equation is described as follows:

$$\forall j \in [0,7], \exists \mathbf{V}_j \in VertexSet_j, Dot(\mathbf{N}_j, \mathbf{V}_i - \mathbf{V}_j) > 0 \tag{1}$$

where $\mathbf{V}_i$ is the center coordinate of the marine entity i. $j \in [0,7]$ is the number of each face of one Octahedral Block. $VertexSet_j$ is the set of vertices of face j. $\mathbf{V}_j$ belongs to the $VertexSet_j$. $\mathbf{N}_j$ is the normal vector of each face.

Additionally, we create an entity information table and a color number for each Octahedral Block. We update the entity information table of the examples as the Octahedral Block's table.

3.2. Marine Cube set synthesis

Inspired by Wang Cubes (Wang, 1961), we cut the Octahedral Blocks into cubes, defined as *Marine Cubes*. Each Marine Cube has six faces: up (down), front (back) and left (right). After randomly choosing six Octahedral Blocks in the corresponding direction (type) of the Octahedral Block set and placing them together and cutting off the extra half of the Octahedral Blocks in each direction, we obtained a Marine Cube. There are also two types of Marine Cube sets: non-submarine Cube set and submarine Cube set. To synthesis a non-submarine Cube, we separately select Octahedral Blocks from UBlocks, FBlocks and LBlocks for its six faces. To synthesis a submarine Cube, we separately select Octahedral Blocks from UBlocks, GFBlocks and GLBlocks for its six faces. In addition, all entities in the bottom half of a submarine Cube should be removed to obtain a ground plane. We create an entity information table for each Marine Cube based on the entity information table of the Octahedral Block, paying particular attention to updating the coordinates of each entity from Octahedral Block's table to Marine Cube's.

During the Synthesis process, each face of a Marine Cube can inherit the color number of the corresponding Octahedral Block (see Section 3.1). We denote the color number of the up, down, front, back, left, and right directions of Marine Cubes as u, d, f, b, l, r, respectively. Each Marine Cube has a number (denoted as Num_{cube}) calculated from the color number. To ensure that a matching Marine Cube is found in each case, we arrange all the color numbers in the Octahedral Block set of every position using the full permutation method. Assuming that each Octahedral Block set contains n Octahedral Blocks, there are n^6 Marine Cubes in the non-submarine Cube set and n^5 Marine Cubes in the submarine Cube set after full permutation. In this paper, the base-n number system is used to store all cases. The conversion equation between color sequence and Num_{cube} is:

$$Num_{cube} = \begin{cases} ((r,l,b,f,u)_n)_{10} & cube \in C_{submarine} \\ ((r,l,b,f,d,u)_n)_{10} & cube \in C_{non\text{-}submarine} \end{cases} \tag{2}$$

where (r,l,b,f,d,u) and (r,l,b,f,u) represent the color sequence in the base-n number system of a non-submarine Cube and a submarine Cube, respectively. $C_{submarine}$ and $C_{non\text{-}submarine}$ represent the submarine Cube set and non-submarine Cube set, respectively. Num_{cube} is a number in the decimal system which corresponds to the color sequence.

3.3. Scene tiling

Next, we tile a marine scene with the Marine Cube sets. Adjacent surfaces of adjacent Marine Cubes with the same color can be combined together. We tile Marine Cubes from down to up, left to right, and back to front in the process of tiling. We define the down, front, and left positions of the current Marine Cube as pre-order Marine Cubes. We only match the pre-order Marine Cubes when tiling. If the pre-order Marine Cubes already exist, the color of its corresponding position is obtained. Otherwise, the color for that direction is selected directly from the Octahedral Block set in the corresponding direction.

In the process of tiling, let (i,j,k) be the index of each Marine Cube in the scene, where i is the left–right direction, j is the up-down direction, and k is the front–back direction. When $j = 0$, the current Marine Cube is selected in the submarine Cubes set, and the corresponding color sequence of the selected Marine Cube is (r,l,b,f,d,u). When $j \neq 0$, the current Marine Cube is selected in the non-submarine Cube set, the corresponding color sequence of the selected Marine Cube is (r,l,b,f,u). The color sequence is calculated as:

$$dir = \begin{cases} Random, dir \in \{u,f,r\} \\ l_{(i,j,k)} = \begin{cases} Random & i = 0 \\ l_{(i-1,j,k)} & i \neq 0 \end{cases} \\ d_{(i,j,k)} = \begin{cases} nonentity & j = 0 \\ d_{(i,j-1,k)} & j \neq 0 \end{cases} \\ b_{(i,j,k)} = \begin{cases} Random & k = 0 \\ b_{(i,j,k-1)} & k \neq 0 \end{cases} \end{cases} \tag{3}$$

where dir is the color number of a cube, and $dir \in \{u,d,f,b,l,r\}$. i,j,k is the index of a cube in scene. $Random$ is a random number from 0 to n. n is the quantity of Octahedral Blocks contained in one Octahedral Block set (see Section 3.1).

3.4. Species and orientation randomization

During the scene tiling process, we use many Marine Cubes. However, due to the limited number of Octahedral Block sets, the distribution and orientation of entities and entity groups may be repeated in the scene. We randomly assign species types to each entity and assign a random orientation to each entity group to solve this problem. When making the Octahedral Blocks, we assigned group numbers to each entity. But during the synthesis

of the Marine Cubes, the same Octahedral Block is used many times, and the group numbers are not unique. Therefore, we need to use the unique index of Marine Cube to update the group number for each entity. Then we assign random species and random orientations based on each entity's group number and species type and update those information into the table of each Marine Cube in the scene. Species used for the random assignment are derived from the library of marine entities. The library of the marine entities is shown in Fig. 5.

3.5. Density adjustment

In the process of tiling, some Octahedral Blocks may have many groups, leading to an overstocking of groups in the final result. In 2D, density control is generally implemented with density maps. In 3D, it is obvious that density control cannot be achieved by one or several 2D density maps, and users cannot quickly obtain 3D density function expressions. To this end, we propose a point-set based density control method to improve user operability and achieve the effect of scene density control at the same time. The factors that affect the survival of the entities in the scene are not only the scene density but also the survival probability and survival depth of the entities. Therefore, we propose a rendering probability formula for the adjustment of scene density. The core idea of this formula is to perform a probability calculation for each marine entity and then decide whether to discard the entity or not. The rendering probability formula is:

$$P(i) = \omega_d * f_d(i) + \omega_u * f_u(i) + \omega_s * s_i \tag{4}$$

where s_i is the survival probability of entity i recorded in entity information table, f_d is the depth probability function, and f_u is the user probability function. $P(i)$ is jointly determined by f_d, f_d and s_i, ω_d, ω_u and ω_s are their respective weights, and the sum of all ω is 1.

The depth probability function $f_d(i)$ is defined as:

$$f_d(i) = K_0 * |d_i - d| \tag{5}$$

where d_i is the current depth of entity i in the scene, d is the survival depth in the entity information table, and K_0 is the decay factor.

The user probability function f is defined as:

$$f_{dis}(dis) = K_1 * Min(|\mathbf{V_i} - \mathbf{V_j}|), j \in U \tag{6}$$

where U is the set of density control points provided by the user, V_i denotes the 3D coordinates of entity i, and V_j is the 3D coordinates of a point j in U. $Min(|V_i - V_j|)$ denotes the minimum distance value of entity i to U, and K_1 is the decay factor. Finally, each entity i is assigned a random number ζ from 0 to 1. When $\zeta > P(i)$, the entity i is discarded.

4. Results and analysis

4.1. Implementation and performance

To validate our method, we choose Unity® software[1] as a desktop environment and implement an interactive authoring application on a desktop PC with an Intel I7 processor, 16G of RAM, GTX 1080Ti GPU. We pre-define three categories of marine entities for creating examples, including fish, marine plants, and reefs. Each category contains twenty species of fish, nine marine submerged plants, and five species of reefs, respectively. Six examples are made for each direction of the Octahedral Blocks, and two Octahedral Blocks are shown in Fig. 4. Fig. 6 shows a subset of the final results authored using our application and Fig. 7 shows the results of our application with and without density adjustment which is presented in Section 3.5. More results are available in the supplementary materials. Meanwhile, we list all the parameters of our experiments and their default settings in Table 1.

Table 1
Parameters of our method in Section 3.5. We use these parameters to obtain the following results.

Parameter	Value	Description
ω_d	0.3	Impact factor of survival depth
ω_{dis}	0.2	Impact factor of density control point set
ω_s	0.5	Impact factor of survival probability
K_0	0.01	Attenuation factor of survival depth
K_1	0.05	Attenuation factor of survival probability

Besides this, we test the running time of our method on the same desktop PC mentioned above. The performance tests focus on the four steps of the approach in this paper: Marine Cube set synthesis (Section 3.2), scene tiling (Section 3.3), species and orientation randomization (Section 3.4), and density adjustment (Section 3.5). Since the first step (Octahedral Blocks creation, Section 3.1) is pre-processed and manually produced, we conduct a user study to evaluate the crafting consumption time of this step instead of directly evaluating its running time (see Section 4.3 for more details). Note that the time spent on the rendering processes, such as drawing meshes and shading, are not key parts of our method, so we ignore the performance tests for these processes. The results of the running time test are shown in Table 2, where the first column of the table indicates that I, J, and K Marine Cubes are used to fill the length, width, and height of the virtual marine scene, respectively. From this table, it can be seen that our method takes only 6.593 s to generate a scene of a scale with $6 \times 6 \times 6$, indicating that the method can effectively support user requirements for quick editing and adjustment and is suitable for adoption in interactive authoring tools.

In particular, since the step of Octahedral Blocks creation (Section 3.1) affects the step of Marine Cube set synthesis (Section 3.2), we also conduct a further performance test on Section 3.2 independently. In this step, the main factors affecting the running speed are the number of Octahedral Block candidates N in each direction, the number of entities M_1^i in each UBlocks, LBlocks, and FBlocks (denoted as i), and the number of entities M_2^j in each GLBocks and GUBlocks (denoted as j). The time for generating Marine Cube set is estimated by $\sum_{i=1}^{6*N} M_1^i * N^5 * t + \sum_{i=1}^{N} M_1^i * N^4 * t + \sum_{j=1}^{4*N} M_2^i * N^4 * t$ (The time taken for an entity to test is t). The size of the Marine Cube set is $N^6 + N^5$ (i.e., the size of the non-submarine Cube set is N^6, and the size of the submarine Cube set is N^5). M depends on the properties of the user-made Octahedral Blocks, so we fix the range of values of M in our test: the number of entities in the non-submarine Cube set ranges from $M_1 \in [100, 200]$ in each example, and the number of entities the submarine Cube set ranges from $M_2 \in [20, 30]$. The effect of different M, N, T on the running time is shown in Table 3. It can be seen that an increase in N will consume more running time. However, since an increase in N can lead to richer synthesis results, a trade-off can be sought in practical use.

[1] https://unity.com/

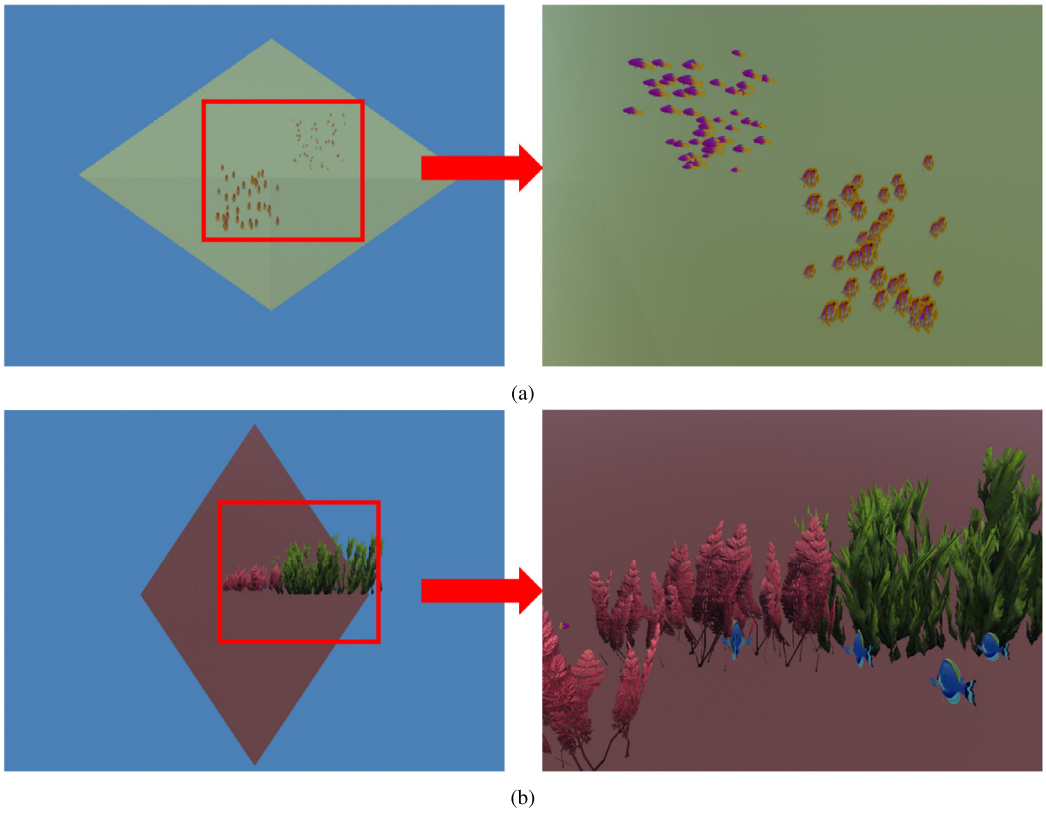


Fig. 4. Illustrations of two Octahedral Blocks selected from UBlocks and GLBlocks, respectively. (a) A block with fish schools. (b) A block with both fish schools and marine submerged plants.

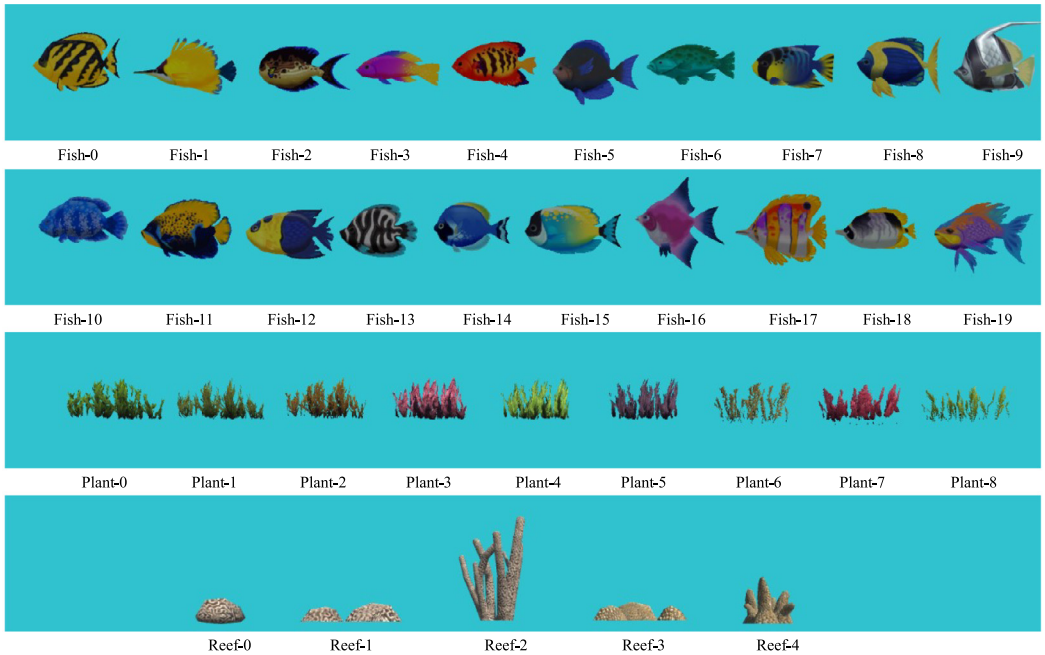
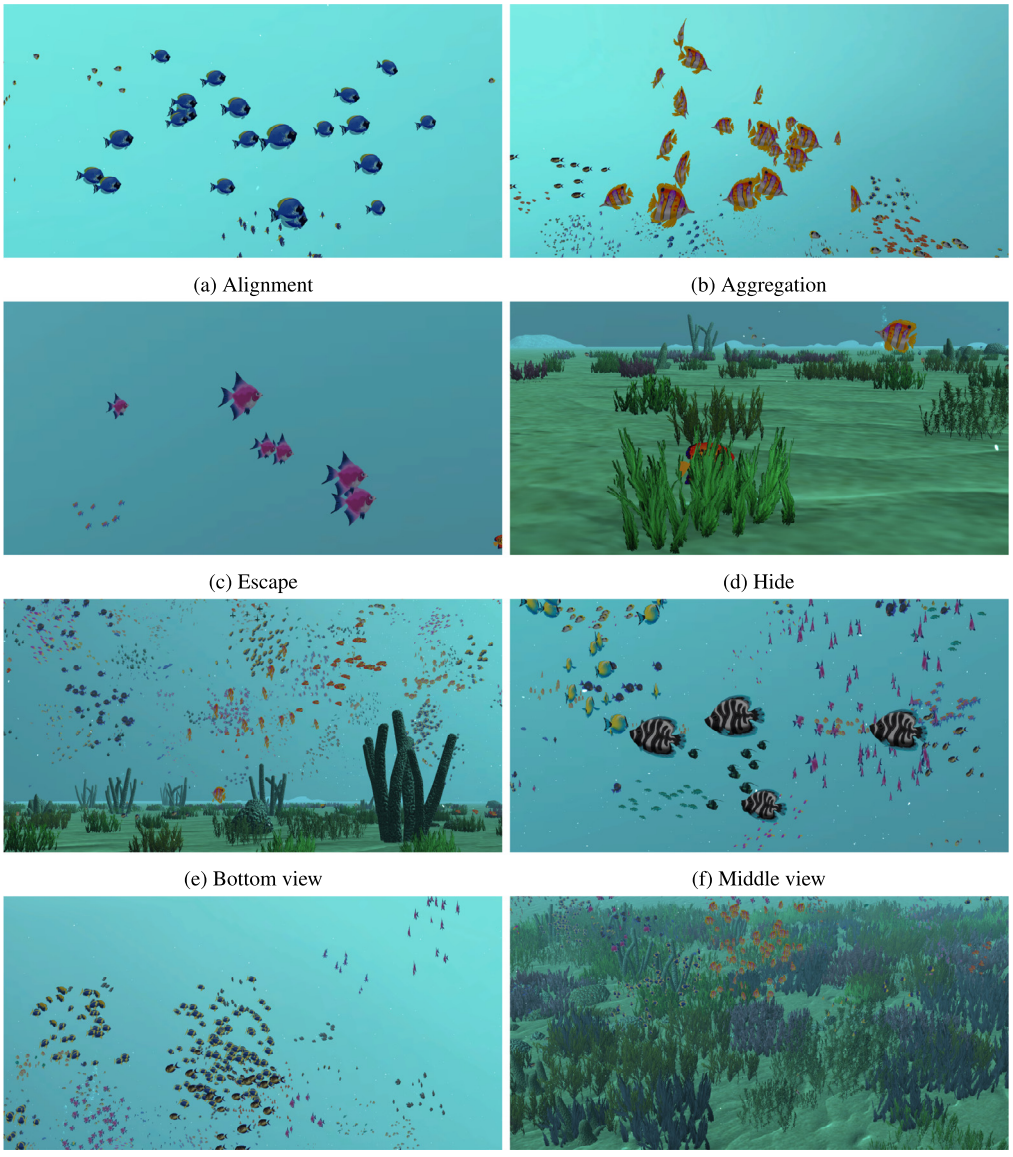


Fig. 5. Library of marine entities. We adopt these marine models for the implementation part of Section 4.1, including twenty types of fish, nine types of marine submerged plants, and five types of reefs.

4.2. Quantitative and visual comparison

Next, to quantitatively assess our method and objectively compare our method to other baseline algorithms, we compare against two group simulation methods: the Boids method (Reynolds, 1987), and the Random method (groups are completely randomly generated, i.e., all the entity groups are randomly generated and randomly placed in the scene). For our method, we implement two versions, one with density adjustment (Section 3.5) and one without density adjustment. We also



(a) Alignment (b) Aggregation

(c) Escape (d) Hide

(e) Bottom view (f) Middle view

(g) Boundary view (h) Seafloor view

Fig. 6. Results of the proposed method: multiple shots of a marine scene simulated in a cube scale of 6 × 6 × 6. (a)–(d) Close-up views of diverse behaviors of marine groups, including alignment, aggregation, escape and hide. (e)–(h) Shots of different areas in the scene.

(a) Our application without density adjustment (b) Our application with density adjustment

Fig. 7. A comparison between our application with and without density adjustment, (a) Our application without density adjustment. (b) Our application with density adjustment.

adopt two comparison metrics, which are proposed by Yang et al. (2020): position deviation and orientation deviation. These two metrics are developed by their reference to the three major rules of the Boids method. Concretely, the position deviation

Table 2
Consuming time of each procedure (in seconds).

Scene scale	Number of entities	Time of Section 3.2	Time of Section 3.3	Time of Section 3.4	Time of Section 3.5	Total
$2 \times 2 \times 2$	552		0.003	0.002	0.001	6.572
$4 \times 4 \times 4$	4,401		0.004	0.002	0.001	6.573
$6 \times 6 \times 6$	13,213	6.566	0.006	0.003	0.001	6.576
$8 \times 8 \times 8$	32,227		0.011	0.004	0.001	6.582
$10 \times 10 \times 10$	58,267		0.020	0.006	0.001	6.593

Table 3
Consuming time of generating Marine Cube set (in seconds).

N	Range of M_1	Range of M_2	T $(N^6 + N^5)$	Time
2			96	0.027
3			976	0.169
4	100∼200	20∼30	5,120	0.729
5			18,750	2.345
6			54,432	6.566

Table 4
Quantitative Comparison of different methods for generating fish schooling behaviors (The closer the range is to the Boids method, the better).

Method	Set of position deviation $\mathbb{R}_1$	Set of orientation deviation $\mathbb{R}_2$
Boids method (Reynolds, 1987)	0.7∼3.0	2∼180
Random method	0.2∼1.5	2∼10
Ours (w/o density adjustment)	0.8∼2.2	10∼104
Ours (with density adjustment)	0.9∼2.2	2∼180

is designed to measure the aggregation behavior and collision avoidance behavior. We calculated the position deviation for each group and obtained the set of positional deviations for all groups in the scene (denoted as $\mathbb{R}_1$). For each group, the position deviation is calculated by:

$$\begin{cases} e_1 = \dfrac{1}{C}\displaystyle\sum_{i=1}^{C} \|\mathbf{p_i} - \bar{\mathbf{p}}\|_2 \,, \forall e_1 \in \mathbb{R}_1 \\ \bar{\mathbf{p}} = \dfrac{1}{C}\displaystyle\sum_{i=1}^{C} \mathbf{p_i}, \end{cases} \tag{7}$$

where e_1 represent the position deviation for one group, C is the total number of individuals in the group, $\mathbf{p_i}$ is the position vector of individuals in the group, and $\bar{\mathbf{p}}$ is the sum of the position vectors of each individual in the group divided by the total number of individuals.

The orientation deviation is designed to measure the isotropic behavior. We calculated the orientation deviation for each group and obtained the set of orientation deviations for all groups in the scene (denoted as $\mathbb{R}_2$). For each group, the orientation deviation is calculated by:

$$\begin{cases} e_2 = \dfrac{1}{C}\displaystyle\sum_{i=1}^{C} \|\mathbf{o_i} - \bar{\mathbf{o}}\|_2 \,, \forall e_2 \in \mathbb{R}_2 \\ \bar{\mathbf{o}} = \dfrac{1}{C}\displaystyle\sum_{i=1}^{C} \mathbf{o_i}, \end{cases} \tag{8}$$

where e_2 is the orientation deviation for one group, C is the total number of individuals in the group, $\mathbf{o_i}$ is the direction vector of individuals in the group, and $\bar{\mathbf{o}}$ is the sum of direction vectors of each individual in the group divided by the total number of individuals.

The comparison results of the four methods (the Boids method (Reynolds, 1987), the Random method, our method with density adjustment, and our method without density adjustment) to measure position deviation and orientation deviation metrics are shown in Table 4. The significance of these two metrics is that different fish schools will maintain a stable position deviation, but the orientation deviation will show a large variation with different fish school shapes (Yang et al., 2020) and thus can be used to measure the local behavior of the fish schools. From the comparison results, it can be seen that the result of our method is similar to the Boids method in both position deviation and orientation deviation and outperforms the Random method (the close to Boids (Reynolds, 1987), the better). It shows that the method in this paper is better than the Random method in terms of local behavior creation and is competitive with the Boids method.

In addition, we also compare the visual results of the four methods mentioned above. For each set of comparisons, a real image is provided as a reference. The comparison results are shown in Fig. 9. Since the Boids method is rule-based, the fish schools generated using this method may follow the same rule, and many fish schools show similar behavior. The fish schools generated from the Random method tend to be too "dull" to match good visual effects. In our method, when no density adjustment is performed, some of the generated fish schools are overstocked. However, after density adjustment is performed, this problem is solved.

4.3. User studies

We have designed two user studies to verify the usability and visual experience of our approach objectively. We designed two tasks for 20 participants (12 males and 8 females) in the age range from 20 to 40, including teachers, university students, and company employees.

For the first task, we expected to evaluate the usability of this method. We invited these 20 participants who had different artistic backgrounds and areas of knowledge for this case to create the Octahedral Blocks. At the beginning of the study, each of them was given a desktop PC and a 10-minute tutorial on our system. Table 5 shows the range of total time spent by participants to create examples and Octahedral Blocks. As can be seen from the table, the time of creating an example of each participant is between 63 and 158 s, and creating an Octahedral Block is between 33 and 97 s. Once the examples are created, the user only needs to interact with a few parameters, and all subsequent steps are automatically computed. This user study demonstrates our method is a very easy-to-use and efficient tool.

Table 5
User Study 1: The range of total time spent by participants to create examples and Octahedral Blocks (in seconds).

Task	Time
Create Examples	63∼158
Create Octahedral Blocks	33∼97

For the second task, we expected to evaluate the visual experience of the results generated by this method. We set up 20 sets of

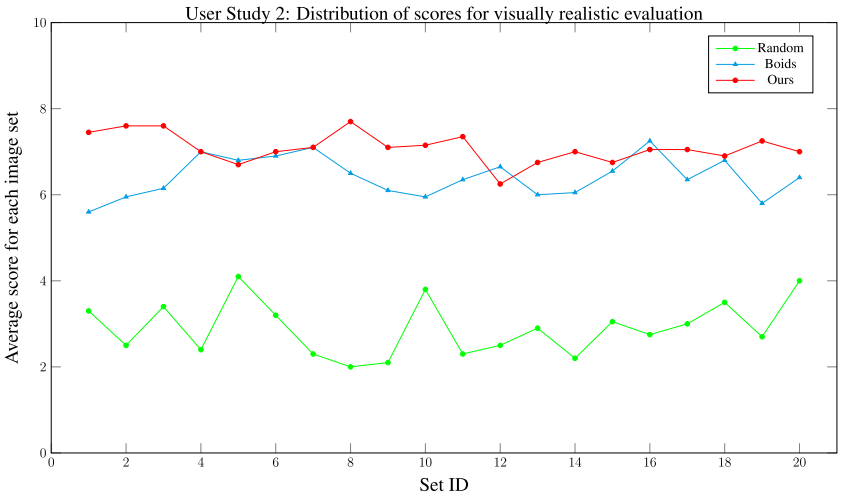
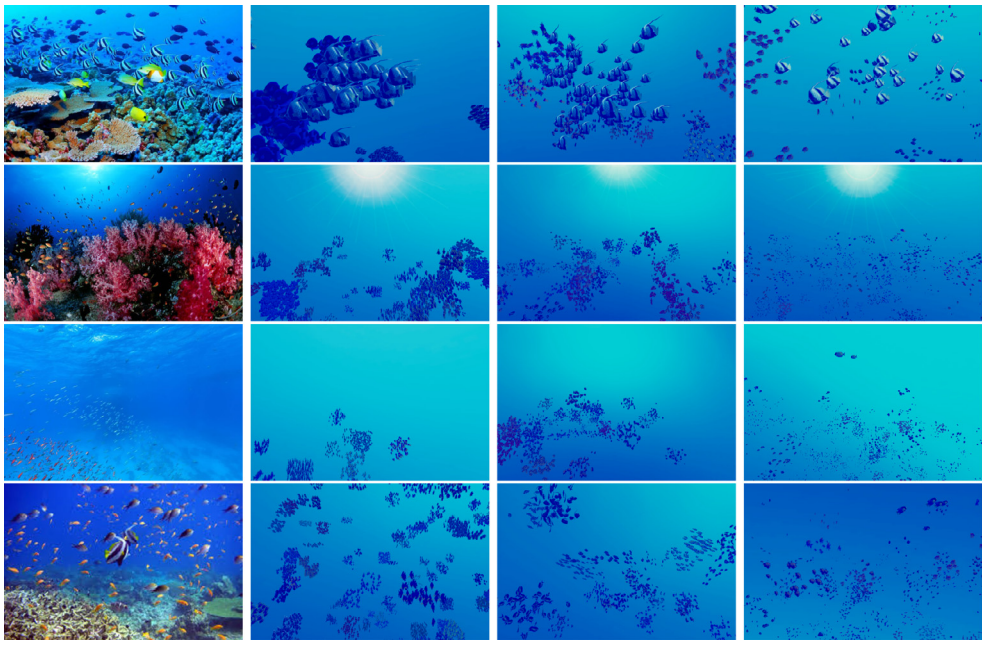


Fig. 8. Questionnaire scores. Overall, the visual quality of our method is better than the Boids method (Reynolds, 1987) and the Random method. The Cronbach α of this user study is 0.983, which indicates that the user study has a high reliability level.



(a) Real (b) Random (c) Boids (d) Ours

Fig. 9. Visual comparison. We provide four sets of images, each providing one real image and three images generated by the Random method, the Boids (Reynolds, 1987) method, and our method. The comparison shows that the Random method appears to be stacked, the Boids method appears to have similar behavior of multiple fish schools, and our method is closer to the real images.

images, each containing 4 images, with a real marine scene image as the reference image and 3 virtual marine images generated by 3 methods (randomly scrambled), including the Boids method, Random method, and Our method. All the virtual marine scene images had similar lighting and camera angles with the corresponding real marine scene image to eliminate additional effects on the visual experience. In addition, since the Boids method cannot generate schools of fish along with marine submerged plants or reefs, this user study only focuses on the fish schools authoring results. We designed a survey questionnaire of a 10-point scale, with 1 being the lowest and 10 being the highest, and asked these 20 participants to rate the 3 virtual marine images in each image set based on how similar the distribution of fish schools was to the corresponding one real image. The evaluation results are shown in Fig. 8, where the X-axis indicates the ID of each image set and the Y-axis indicates the average score of each image in each set. We conducted a reliability test on the results of this questionnaire, which had a Cronbach's α coefficient of 0.983, indicating that the questionnaire has high data reliability. The overall average scores for all image sets were: 6.41 for the Boids method, 3.15 for the Random method, and 7.08 for Ours, indicating that the results generated by the method are in accord with human preferences. Specifically, the table shows that our method scored higher than the Random method in each of the 20 sets of images and higher than the Boids method in 15 sets. Among the 20 sets of images, we designed 10 image sets with full scene views (set numbers: 1, 9, 10, 12, 15, 11, 16, 17, 18, 19, 20), which all contain more than 10 fish, to measure the overall scene effect. In the scoring results of these sets, our method scored higher than the Boids method in 8 sets, which indicates that our method is better in the overall effect of scene authoring results; we also designed 10 image sets with single fish school views (set

numbers: 2, 3, 4, 5, 6, 7, 8, 11, 13, 14), which focus on a single fish school and are used to measure the local fish behaviors. In the scoring results of these sets of images, our method does not differ much from the Boids' scores, which indicates that we can match the Boids method in terms of the effect of fish behavior details, which is also consistent with the experimental results in Section 4.2.

4.4. Limitations

Despite the good results obtained, our approach still has some limitations. We leave these limitations to the open discussion of future work:

- Our method can only author static marine scenes and is therefore only suitable for creating the initial state of the marine groups.
- Creating Octahedral Blocks cannot deal with individuals in the input examples with sizes larger than an Octahedral Block.

5. Conclusions

In this paper, we have proposed a fast method for authoring marine scenes using Wang Cubes. The method is able to create large-scale marine scenes by extracting information about the real distribution of marine groups of entities from real photos to enhance the realism and accuracy of simulation results. During the authoring steps, users can customize the entity information table and library of marine entities and use parameters to adjust the density. The method uses Wang Cubes to combine Octahedral Blocks into Marine Cubes, which enriches the diversity of marine group distribution. In the experimental part, we validated the method's performance and compared it quantitatively and visually with four methods. Finally, we designed two user studies to demonstrate that our method is able to create diverse marine scenes that match human preferences in an efficient way.

It is worth mentioning that although the marine scene is selected as our research subject, the method proposed in this paper is generic and it allows for fast authoring of any 2D and 3D mixed groups (crowds). For example, our method can also author scenes such as a forest with bird flocks, farmland infested by locust swarms, a city park inhabited by pigeons, etc.

CRediT authorship contribution statement

Siyuan Zhu: Methodology, Software, Writing – original draft. **Xinjie Wang:** Conceptualization, Writing – review & editing, Funding acquisition. **Ming Wang:** Validation, Investigation. **Yucheng Wang:** Validation, Data curation. **Zhiqiang Wei:** Project administration. **Bo Yin:** Supervision. **Xiaogang Jin:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval

This study does not contain any studies with human or animal subjects performed by any of the authors. All data used in the study are taken from public databases that were published in the past.

Acknowledgments

Xinjie Wang was supported by the Shandong Provincial Natural Science Foundation of China (Grant No. ZR2021QF124), China Postdoctoral Science Foundation (Grant No. 2021M703031), Qingdao Postdoctoral Applied Research Foundation, China, and the Open Project Program of the State Key Lab of CAD&CG (Grant No. A2219), Zhejiang University. Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant No. 62036010). Yucheng Wang was supported by the National Key Research and Development Program of China (Grant No. 2020YFB0204804).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.visinf.2022.05.004.

References

Alaliyat, S., Yndestad, H., Sanfilippo, F., 2014. Optimisation of boids swarm model based on genetic algorithm and particle swarm optimisation algorithm (comparative study). In: Proceedings of European Council for Modeling and Simulation ECMS 2014. pp. 643–650, URL http://dx.doi.org/10.7148/2014-0643.

Berger, R., 1966. The Undecidability of the Domino Problem. American Mathematical Society, URL http://dx.doi.org/10.1007/978-3-030-57666-0_6.

Bi, H., Mao, T., Wang, Z., Deng, Z., 2016. A data-driven model for lane-changing in traffic simulation. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 149–158.

Chao, Q., Deng, Z., Ren, J., Ye, Q., Jin, X., 2017. Realistic data-driven traffic flow animation using texture synthesis. IEEE Trans. Vis. Comput. Graphics 24 (2), 1167–1178, URL http://dx.doi.org/10.1109/TVCG.2017.2648790.

Chao, Q., Shen, J., Jin, X., 2013. Video-based personalized traffic learning. Graph. Models 75 (6), 305–317, URL http://dx.doi.org/10.1016/j.gmod.2013.07.003.

Chen, Q., Luo, G., Tong, Y., Jin, X., Deng, Z., 2019. Shape-constrained flying insects animation. Comput. Animat. Virtual Worlds 30 (3–4), e1902, URL http://dx.doi.org/10.1002/cav.1902.

Chen, Y.-R., Zhang, X.-X., Yu, Y.-S., Ma, S.-W., Yang, B., 2021. Enhancing convergence efficiency of self-propelled agents using direction preference. Physica A 586, 126415, URL http://dx.doi.org/10.1016/j.physa.2021.126415.

Cohen, M.F., Shade, J., Hiller, S., Deussen, O., 2003. Wang tiles for image and texture generation. ACM Trans. Graph. 22 (3), 287–294, URL http://dx.doi.org/10.1145/882262.882265.

Culik, K., Kari, J., 1996. An aperiodic set of wang cubes. In: J.UCS J. Univ. Comput. Sci.. Springer Berlin Heidelberg, pp. 675–686, URL http://dx.doi.org/10.1007/978-3-642-80350-5_57.

Derouet-Jourdan, A., Mizoguchi, Y., Salvati, M., 2016. Wang tile modeling of wall patterns. In: Mathematical Progress in Expressive Image Synthesis III. Springer, pp. 71–81, URL http://dx.doi.org/10.1007/978-981-10-1076-7_9.

Fisher, M., Ritchie, D., Savva, M., Funkhouser, T., Hanrahan, P., 2012. Example-based synthesis of 3D object arrangements. ACM Trans. Graph. 31 (6), 1–11, URL https://doi.org/10.1145/2366145.2366154.

Ginelli, F., 2016. The physics of the vicsek model. Eur. Phys. J. Spec. Top. 225 (11), 2099–2117, URL http://dx.doi.org/10.1140/epjst/e2016-60066-8.

Hartman, C., Benes, B., 2006. Autonomous boids. Comput. Animat. Virtual Worlds 17 (3–4), 199–206, URL http://dx.doi.org/10.1002/cav.123.

Helbing, D., Buzna, L., Johansson, A., Werner, T., 2005. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transp. Sci. 39 (1), 1–24, URL http://dx.doi.org/10.1287/trsc.1040.0108.

Iizuka, H., Nakamoto, Y., Yamamoto, M., 2018. Learning of individual sensorimotor mapping to form swarm behavior from real fish data. In: ALIFE 2018: The 2018 Conference on Artificial Life. pp. 179–185, URL http://dx.doi.org/10.1162/isal_a_00039.

Inomata, Y., Takami, T., 2020. Analysis of the collective behavior of boids. In: Traffic and Granular Flow 2019. Springer, pp. 373–379, URL https://doi.org/10.1007/978-3-030-55973-1_46.

Ju, E., Choi, M.G., Park, M., Lee, J., Lee, K.H., Takahashi, S., 2010. Morphable crowds. ACM Trans. Graph. 29 (6), 1–10, URL http://dx.doi.org/10.1145/1882261.1866162.

Karamouzas, I., Sohre, N., Narain, R., Guy, S.J., 2017. Implicit crowds: Optimization integrator for robust crowd simulation. ACM Trans. Graph. 36 (4), 1–13, URL http://dx.doi.org/10.1145/3072959.3073705.

Kim, S., Bera, A., Best, A., Chabra, R., Manocha, D., 2016. Interactive and adaptive data-driven crowd simulation. In: 2016 IEEE Virtual Reality. pp. 29–38, URL http://dx.doi.org/10.1109/VR.2016.7504685.

Lee, K.H., Choi, M.G., Hong, Q., Lee, J., 2007. Group behavior from video: a data-driven approach to crowd simulation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 109–118.

Li, W., Wolinski, D., Lin, M.C., 2017. City-scale traffic animation using statistical learning and metamodel-based optimization. ACM Trans. Graph. 36 (6), 1–12, URL http://dx.doi.org/10.1145/3130800.3130847.

Lu, A., Ebert, D.S., Qiao, W., Kraus, M., Mora, B., 2007. Volume illustration using wang cubes. ACM Trans. Graph. 26 (2), 11–es, URL https://doi.org/10.1145/1243980.1243985.

Narain, R., Golas, A., Curtis, S., Lin, M.C., 2009. Aggregate dynamics for dense crowd simulation. In: ACM SIGGRAPH Asia 2009 Papers. Association for Computing Machinery, pp. 1–8, URL http://dx.doi.org/10.1145/1661412.1618468.

Paranjape, A.A., Chung, S.-J., Kim, K., Shim, D.H., 2018. Robotic herding of a flock of birds using an unmanned aerial vehicle. IEEE Trans. Robot. 34 (4), 901–915, URL http://dx.doi.org/10.1109/TRO.2018.2853610.

Ren, J., Wang, X., Jin, X., Manocha, D., 2016. Simulating flying insects using dynamics and data-driven noise modeling to generate diverse collective behaviors. PLoS One 11 (5), e0155698, URL http://dx.doi.org/10.1371/journal.pone.0155698.

Ren, J., Xiang, W., Xiao, Y., Yang, R., Manocha, D., Jin, X., 2021. Heter-sim: Heterogeneous multi-agent systems simulation by interactive data-driven optimization. IEEE Trans. Vis. Comput. Graphics 27 (3), 1953–1966, URL http://dx.doi.org/10.1109/TVCG.2019.2946769.

Reynolds, C.W., 1987. Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. pp. 25–34, URL http://dx.doi.org/10.1145/37401.37406.

Shen, J., Wang, X., Chao, Q., Jin, X., 2014. Fast and large-scale crowd authoring based on samples. Chinese J. Comput. 37 (3), 621–631.

Sibley, P.G., Montgomery, P., Marai, G.E., 2004. Wang cubes for video synthesis and geometry placement. In: ACM SIGGRAPH 2004 Posters. Association for Computing Machinery, p. 20, URL http://dx.doi.org/10.1145/1186415.1186439.

Silva, A.R.D., Lages, W.S., Chaimowicz, L., 2010. Boids that see: Using self-occlusion for simulating large groups on gpus. Comput. Entertain. 7 (4), 1–20, URL http://dx.doi.org/10.1145/1658866.1658870.

Snape, J., Van Den Berg, J., Guy, S.J., Manocha, D., 2011. The hybrid reciprocal velocity obstacle. IEEE Trans. Robot. 27 (4), 696–706, URL http://dx.doi.org/10.1109/TRO.2011.2120810.

Stam, J., 1997. Aperiodic Texture Mapping. European Research Consortium for Informatics and Mathematics.

Treuille, A., Cooper, S., Popović, Z., 2006. Continuum crowds. ACM Trans. Graph. 25 (3), 1160–1168, URL http://dx.doi.org/10.1145/1141911.1142008.

Tu, X., Terzopoulos, D., 1994. Artificial fishes: Physics, locomotion, perception, behavior. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques. pp. 43–50, URL http://dx.doi.org/10.1145/192161.192170.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., Shochet, O., 1995. Novel type of phase transition in a system of self-driven particles. Phys. Rev. Lett. 75 (6), 1226–1229, URL http://dx.doi.org/10.1103/PhysRevLett.75.1226.

Wang, H., 1961. Proving theorems by pattern recognition—II. Bell Syst. Tech. J. 40 (1), 1–41, URL https://doi.org/10.1002/j.1538-7305.1961.tb03975.x.

Wang, X., Jin, X., Deng, Z., Zhou, L., 2014. Inherent noise-aware insect swarm simulation. Comput. Graph. Forum 33 (6), 51–62, URL http://dx.doi.org/10.1111/cgf.12277.

Xiang, W., Yao, X., Wang, H., Jin, X., 2020. FASTSWARM: A data-driven framework for real-time flying insect swarm simulation. Comput. Animat. Virtual Worlds 31 (4–5), e1957, URL http://dx.doi.org/10.1002/cav.1957.

Xu, K., Chen, K., Fu, H., Sun, W.-L., Hu, S.-M., 2013. Sketch2scene: Sketch-based co-retrieval and co-placement of 3D models. ACM Trans. Graph. 32 (4), 1–15, URL https://doi.org/10.1145/2461912.2461968.

Xueying, L., Shi, M., Li, C., 2017. Design and implement of swarm self-organization model based on Unity3D. In: 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference. pp. 146–149, URL http://dx.doi.org/10.1109/ITNEC.2017.8284926.

Yang, H., Huang, W., Ao, F., 2020. Simulation on self-organization behaviors of fish school based on reinforcement learning (in Chinese). J. Natl. Univ. Def. Technol. 42 (1), 194–202, URL http://dx.doi.org/10.11887/j.cn.202001027.

Zhang, S.-H., Zhang, S.-K., Liang, Y., Hall, P., 2019. A survey of 3D indoor scene synthesis. J. Comput. Sci. Tech. 34 (3), 594–608, URL https://doi.org/10.1007/s11390-019-1929-5.

Zhang, S.-H., Zhang, S.-K., Xie, W.-Y., Luo, C.-Y., Yang, Y., Fu, H., 2021. Fast 3D indoor scene synthesis by learning spatial relation priors of objects. IEEE Trans. Vis. Comput. Graphics 1, URL https://doi.org/10.1109/TVCG.2021.3050143.

Zhou, S., Liu, W., Lou, P., 2015. A new artificial fish swarm algorithm. J. Guangdong Univ. Technol. 32 (4), 99–104, URL http://dx.doi.org/10.3969/j.issn.1007-7162.2015.04.018.