Exploring Swarm-Based Visual Effects

Abstract. In this paper, we explore the visual effects of animated 2D line strokes and 3D cubes. A given 2D image is segmented into either 2D line strokes or 3D cubes. Each segmented object (i.e., line stroke or each cube) is initialised with the position and the colour of the corresponding pixel in the image. The program animates these objects using the boid framework. This simulates a flocking behavior of line strokes in a 2D space and cubes in a 3D space. In this implementation the animation runs in a cycle from the disintegration of the original image to a swarm of line strokes or 3D cubes, then the swarm moves about and then integrates back into the original image (an example clip has been uploaded to YouTube and can be viewed at https://www.youtube.com/watch?v=a-V6h0VzTZ8).

Keywords Computer generated visual effects, Swarm-based VFX

1 Background

Early visual effects (VFX) in medias are mostly accomplished using non-digital techniques such as *stop-motion, optical printing, matte painting*, etc. Thanks to the advancement in digital image processing, VFX has now moved to a different level where the limitation is capped only by our imagination. Currently, most of the high-end post-production editing tools provide various VFX facilities. It is undeniable that VFX has become a crucial part of storytelling in games and films [1].

The particle system has been extensively employed in computer generated VFXs where particles abstract objects in the physical world. The objects may have various physical properties (e.g., mass, shape, colour, velocity, viscosity, etc.). By carefully controlling those parameters, the particle system can emulate various natural events such as fire, smoke, clouds, explosion, etc. The particle approach also has successfully modelled stylistic swarm-like movements of animals such as bird flocking and fish schooling. The simulated swarm-like behaviours are emerging behaviours from the interactions of different individual particles. This kind of simulation is useful in investigating collective behaviours such as foraging, escaping, flocking, etc.

In this paper, we explore the potential of particle-based VFXs to control the swarming effect of small image segments. For example, a given 2D image could be segmented into many small tiles. This creates a mosaic rendering of an original image. Each mosaic tile can be programmed to move about in a 3D/2D space. To simulate the swarming effect, these tile particles move according to three forces: separation, alignment and cohesion [2]. Our system registered the

initial location of each mosaic tile. Hence, the system could generate pleasing emerging behaviours of the tiles swarming out from a disintegrated image as well as swarming in to create an original image.

This paper is organised into the following sections: Section 2 gives an overview of related works; Section 3 discusses our proposed concept and gives the details of the techniques behind it; Section 4 provides the output of the proposed approach; and finally, the conclusion and further research are presented in Section 5.

2 Related Work

Arcimboldo produced many portrait paintings by compositing various objects such as vegetables, fruits, animals [3]. This activity creates new emerging meanings from basic primitives. This kind of concept has been widely explored by designers, artists and computer scientists e.g., emergent computing [4, 5]. Here we explore the idea of mosaics where an image emerges from a composition of various coloured tiles viewed from a distance.

Digital mosaics have been explored by many researchers in the past. Researchers have looked into various generative approach as considering the choices of tiles and how the tiles are placed. In its simplest form, an image can be easily converted into photomosaic of square tiles. In a more sophisticated manner, various components in an image can be segmented out first before generating mosaic patterns for each of them [6, 7]. The latter process produces a much more stylistic tile composition.

Mosaics are a static art form. The artefact is displayed when the tiles are fully composed. In this paper, mosaic tiles are animated in a swarm-like movement in a 3D space. We also explore the movement of the line stroke in a swarm-like movement in a 2D space. In both 2D/3D animations, each mosaic tile or line stroke is represented as a particle in a particle swarm.

In [8], the author analysed empirical biological data and proposed a mathematical model to describe the *cohesiveness* of a fish school. Many reserachers have analysed swarm behaviours to gain more understanding of the emergent properties and the shape of the swarm [9, 2, 10].

In computer graphics area, the particle system has been employed to create special effects such as fire, smoke, etc. Researchers have experimented with the idea of particles to generate caricatures [11], flocking and schooling simulations. Perhaps, the simulation by Reynolds [2] has the most impact in this area. He proposed the *bird-oid object* (boid) framework that describes swarm-like behaviours with simple intuitive rules. The boid framework emulates the flocking behaviour with the following basic individual behaviours: (i) avoid crowding particles by ensuring that particles cannot be too close to each other, (ii) steer itself toward the overall direction of the swarm, and (iii) steer itself toward the centre of the swarm. The original boid framework has been extended by other researchers to emulate other swarm behaviours such as splitting and uniting a flock.

Fig. 1. Left: the original image is segmented into many small cubes. Right: these cubes are animated using the boid framework.

Fig. 2. Left: the original image is segmented into many line stroke segments. Right: these line strokes are animated using the boid framework.

3 Materials and Methods

Let $I(h, w)$ be a pixel from row h and column w of an image I having the width and height of h and w pixels. Let p_d be a particle d from a swarm $\mathbf{P}$ of size $|\mathbf{P}| = h \times w$. The image I can be represented using $h \times w$ particles where the

particle p_1 is initialised with the information of the pixel at the location $I(1,1)$, p_2 from $I(1,2)$ and $p_{h\times w}$ from $I(h,w)$.

In this implementation, for a swarm-like visual effect, the size of each particle is bigger than a single pixel. Here, each particle occupies a space of size $m \times n$ pixels. Hence, $\frac{h}{m} \times \frac{w}{n}$ particles are employed to represent the original image. Each particle p_d is an instance of the object class particle, where $p_d.loc()$ returns the location information of the particle d in the 3D space and $p_d.vel()$ returns the velocity information accordingly.

In this paper, the concept of particles was applied to create a visual effect of the swarm movements of many small image segments. Each segment represented a particle in a swarm where its position in the 3D space was controlled using the boid framework. Although these segments were rectangles of image components segmented from the original image, a swarm particle could take any geometrical shape and it would be initialised with the pixels' information of the corresponding segment. In this implementation, the swarm particles took two different types of shapes: cubes and lines. For a cube, all 6 faces were initialised with the pixels' information of the corresponding rectangle segment. The cube particles moved about in a 3D space and the line particles moved about in a 2D space.

3.1 The Boid Framework

The boid framework suggests three important heuristics that compute three velocities $\mathbf{v}_s$, $\mathbf{v}_a$, and $\mathbf{v}_c$ which are attributed to the following properties, respectively: separation, alignment and cohesion respectively. For each particle p_d with the velocity $\mathbf{v}_d$, the steering velocity $\mathbf{v}_d$ can be computed from $\mathbf{v}_d(t+1) = \mathbf{v}_s(t)+\mathbf{v}_c(t)+\mathbf{v}_a(t)-\mathbf{v}_d(t)$. The contributions from $\mathbf{v}_s$, $\mathbf{v}_a$, and $\mathbf{v}_c$ are computed as follow:

$$\mathbf{v}_s = \sum_{i=1}^{|\mathbf{P}|} \mathcal{N}_i k_s (p_d.loc() - p_i.loc())$$

where $\mathcal{N}_i = 1$ if p_i is within the desired neighborhood of p_d, else $\mathcal{N}_i = 0$. The neighborhood of p_d is a sphere of radius r (arbitrarily set by the users) and k_s is a normalisation factor that moderates the effect of the distance $\|p_d - p_i\|$. In other words, $\mathbf{v}_s$ is a vector formed from summation of all $(p_d.loc() - p_i.loc())$ vectors. $\mathbf{v}_a$ and $\mathbf{v}_c$ are computed in the same fashion but with a different normalisation condition:

$$\mathbf{v}_a = \sum_{i=1}^{|\mathbf{P}|} \mathcal{N}_i k_a p_i.vel(); \text{ where } i \neq d$$

and k_a is the normalisation factor.

$$\mathbf{v}_c = \sum_{i=1}^{|\mathbf{P}|} \mathcal{N}_i k_c (p_i.loc() - m_{\mathcal{N}}.loc())$$

where k_c is a normalisation factor that moderates the effect of the distance $\|p_i - m_{\mathcal{N}}\|$. Here $m_{\mathcal{N}}$ is the center of all particles in the neighborhood of p_d.

3.2 The Swarm

Given an input image, a swarm population is generated by segmenting the 2D image into many small segments. Each particle i in the swarm is initialised with a random velocity value $\mathbf{v}_i =< v_x, v_y, v_z >$. At each time step, the swarm behaviour emerges as a result from $\mathbf{v}_s$, $\mathbf{v}_a$, and $\mathbf{v}_c$ discussed in the previous section.

The boundaries front-back (z-axis), top-bottom (y-axis), and left-right (x-axis) are wrapped around. Hence, a particle that moves deep pass the bordering depth will enter the scene again at the frontmost position; a particle that moves out of the top border will emerge from the bottom border; and a particle that moves out of the right border will emerge from the left border and vice versa. This creates a seamless motion of particles.

We implemented three behaviours: (i) flocking behaviour according to the boid framework, (ii) swarming toward a specific target position, and (iii) returning to the original position i.e., forming an original 2D image. Behaviour (i) was obtained by the boid framework. This is an emergent behaviour of the swarm. Behaviour (ii) was obtained by (in each time frame) randomly steering 10% of the swarm population $\mathbf{P}^{10}$ toward the desired target T. The remaining 90% of the swarm population $\mathbf{P}^{90}$ was still controlled by the boid framework.

$$\forall_{i \in \mathbf{P}^{10}} \quad \mathbf{v}_i = k_2(T.loc() - p_i.loc())$$

$$\forall_{i \in \mathbf{P}^{90}} \quad \mathbf{v}_i(t+1) = \mathbf{v}_s(t) + \mathbf{v}_c(t) + \mathbf{v}_a(t) - \mathbf{v}_i(t)$$

Behaviour (iii) was obtained by (in each time frame) steering each particle i back to its original position o_i.

$$\forall_{i \in \mathbf{P}} \quad \mathbf{v}_i = k_3(o_i.loc() - p_i.loc())$$

where k_2 and k_3 were constants that regulated the speed of behaviours (ii) and (iii).

4 Discussion

The purpose of this work is to generate a swarm movement using a particle system. A given image was segmented into many small segments and their pixels' information was used to initialise a swarm of particle; which would be programmed into any desired shape.

We successfully implemented a controller for three different behaviours of the particle swarm. These parameters were controlled by users using shortcut-keys. In this implementation, the default behaviour was the flocking behaviour according to the boid framework. If the user pressed key 'T', then the particle swarmed into two alternate active targets on the bottom left and the right area of the screen. If the user pressed key 'H', then each particle swarmed into its original position. If the user pressed key 'S', then the particles resumed their flocking behaviour.

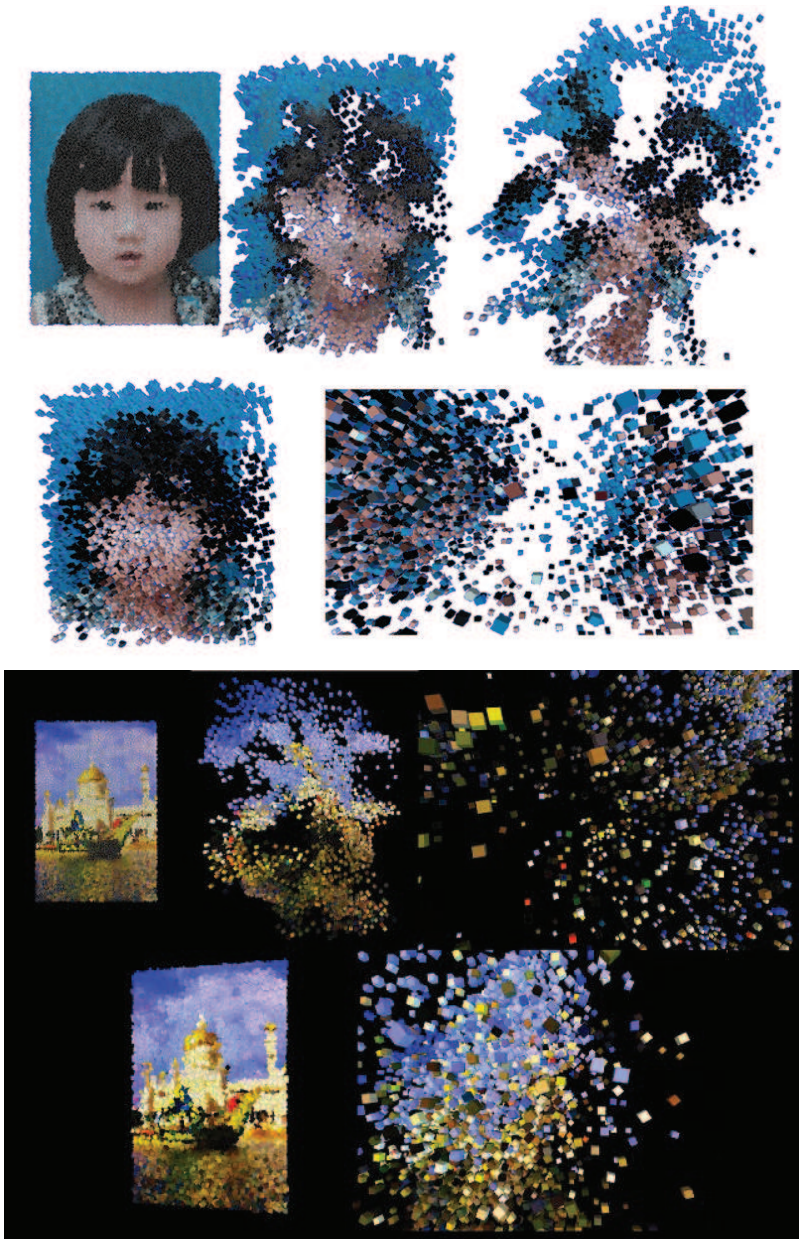

Fig. 3. Ten snap shots from two animation clips, for each clip, from topleft and in a clockwise direction: (i) original image, (ii, iii, iv) the swarm of cubes is simulated, and (v) the swarm integrate back to form the original image.

Fig. 4. Snap shots of animation frames, from topleft and in a clockwise direction: (i) original image, (ii, iii, iv, v) the swarm of cubes is simulated, and (vi) the swarm integrate back to form the original image.

We simulated two swarm appearances: swarm of mosaics and swarm of line strokes (see https://www.youtube.com/watch?v=a-V6h0VzTZ8). Figure 3 shows 10 snapshots (5 from each example) of a swarm of mosaics. The exercise in this figure was motivated by the idea to create a swarm movement of mosaics. An original image disintegrated into a swarm of cubes, traversing in a 3D space before integrating back to its original position. We arbritarily set the width of each cube at 7 pixels as we felt that this produced an appealing swarm movement.

Figure 4 shows 5 snapshots of a swarm of line strokes. This exercise was motivated by the idea to generate a swarm of line strokes that could recombine into drawings or paintings.

5 Conclusion & Future Work

We have shown two examples of swarm-based visual effect: a swarm of cubes and a swarm of lines. The swarming pattern is infinitely complex since each particle is randomly initialised with $< v_x, v_y, v_z >$. The swarming visual FXs are common ingredients in games, films, and advertisements. The footage can be further composited with other materials.

Many interesting areas can be extended from the current work, for examples, stylistic swarm movements, rich particle shapes, etc. In this work, we also explore the use of camera to create various perspective of the swarm. However, the camera positions were manually coded and it was a challenge to select appropriate dynamic camera posoitions. This can be further explored in the future work e.g., to analyse the swarm formation and automatically position the camera at appropriate positions at different times.

References

1. McClean, S.T.: *Digital Storytelling: The Narrative Power of Visual Effects in Film.* The MIT Press, 2008.
2. Reynolds, C.W.: Flocks, heards, and schools: A distributed behavioral model *Computer Graphics, 21*(4):25-34, 1987.
3. Maiorino, G.: *The Portrait of Eccentricity: Arcimboldo and the Mannerist Grotesque*. The Penn State University Press, 1991.
4. Kennedy, J., and Eberhart, R.C., with Shi, Y.: *Swarm Intelligence*. Morgan Kaufmann, 2001.
5. Hastings, E., and Guha, R., and Stanley, K.O.: NEAT particles: Design, representation, and animation of particle system effects In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG07)*. Piscataway, NJ: IEEE, 2007.
6. Kim, J., and Pellacini, F.: Jigsaw image mosaics. In *Proceedings of the* 29^{th} *Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2002*, pp 657-664, 2002.
7. Battiato, S., and Blasi, G.D., and Farinella, G.M. and Gallo, G.: A novel technique for opus vermiculatum mosaic rendering. In *the* 14^{th} *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'06)*. pp 133-140, 2006.
8. Breder, C.M.: Equations descriptive of fish schools and other animal aggregations *Ecology,35*:361-370, 1954.
9. Parrish, J.K., and Viscido, S.V., and Grnbaum, D.: Self-organized fish schools: An examination of emergent properties. *The Biological Bulletin. 202:*296-305, June 2002.
10. Hemelrijk, C.K., and Hildenbrandt, H.: Some causes of the variable shape of flocks of birds. *PLoS ONE 6*(8): e22479. doi:10.1371/journal.pone.0022479, 2011.
11. Phon-Amnuaisuk, S.: Exploring particle-based caricature generations. In *Proceedings of the International Conference on Informatics Engineering and Information Science (ICIES2011)*, pp 37-46, Malaysia, 2011.