Flow Fields and Agents for Immersive Interaction in *Mutator VR: Vortex*

Lance Putnam[1] William Latham[1] Stephen Todd[1]

l.putnam@gold.ac.uk w.latham@gold.ac.uk stephentodd@gmail.com

[1]Computing, Goldsmiths, University of London, United Kingdom

Abstract

This paper discusses the challenges in creating *Mutator VR: Vortex*, a virtual reality experience based on interaction with semi-autonomous, organically-inspired agents. The work allows the immersant to morph between a vast number of procedurally-generated microworlds each with its own visual elements, sounds, agent dynamics, and user interactions. We outline two methods used for procedural generation that are based fundamentally on integration of different modalities. *Curve-based synthesis* is used for simultaneous generation of entity sounds and shape and *flow grains* are employed to determine both agent dynamics and user interaction with the agents.

1 Introduction

Virtual reality (VR) is an apt platform for delivering integrated sensory experiences. Unlike many other media, it has the ability to deliver a more concrete *spatial* experience while still including traditional modalities such as sound, graphics, and interactivity. In fact, one could argue that space or sense of space is the key component of the "reality" in VR, more so than, say, photorealism or other adaptations of physical reality. Space, in turn, can serve to integrate multiple modalities into an experience by acting as a deeper structural element. It seems fitting to develop algorithms and techniques for virtual reality applications that use space as a central underlying concept.

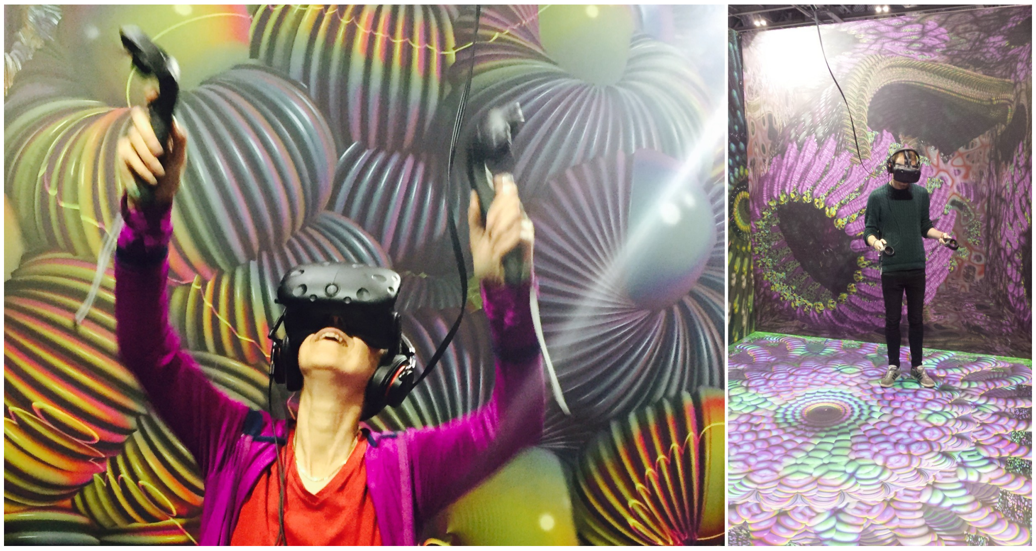
The motivation for this work is to investigate unified approaches to defining the sounds, geometry, and interactive dynamics of a world populated with agents. The criteria for content generation algorithms is that they produce output that is rich, varied, and unpredictable. In addition, space should be central to any algorithm used. Additionally, the experience should have as many emergent aspects as possible with respect to user goals and interactions [Gilbert, 2011] and generated sounds and graphics.

Mutator VR is a VR-based installation that was first exhibited in September 2016 at the New Scientist Live Expo[1] at the ExCel Exhibition Centre in London (Figure 1a) attended by twenty-two thousand visitors and will be shown in forthcoming exhibitions in Korea and the UK. *Mutator VR: Vortex* builds on the Mutator work of Todd and Latham [Todd and Latham, 1992] to give the viewer a new form of immersive interaction in virtual reality beyond the “pick and breed” approach associated with Mutator [Lewis, 2008]. Immersants intervene in and directly affect a swirling mass of agent-based organic forms floating and moving in a large 3D virtual environment (Figure 1b). *Mutator VR: Vortex* gives the viewer a direct sense of presence through cause and effect by attracting or repelling the swirling organic agents. The aim was to achieve a similar sense of interaction as a diver attracting fish with food and to give the illusion that these organic forms are living. One key goal of the interaction is that it be emergent without predetermined goals so that the user has more opportunity to construct their own experience. Emergent interactions may include juggling agents from hand to hand using attractive forces, throwing clusters of agents, or conducting abstract flow patterns. The aim of the work is to deliver an open-ended and playful experience. The challenge is to create a mathematical framework with correct parameter values that

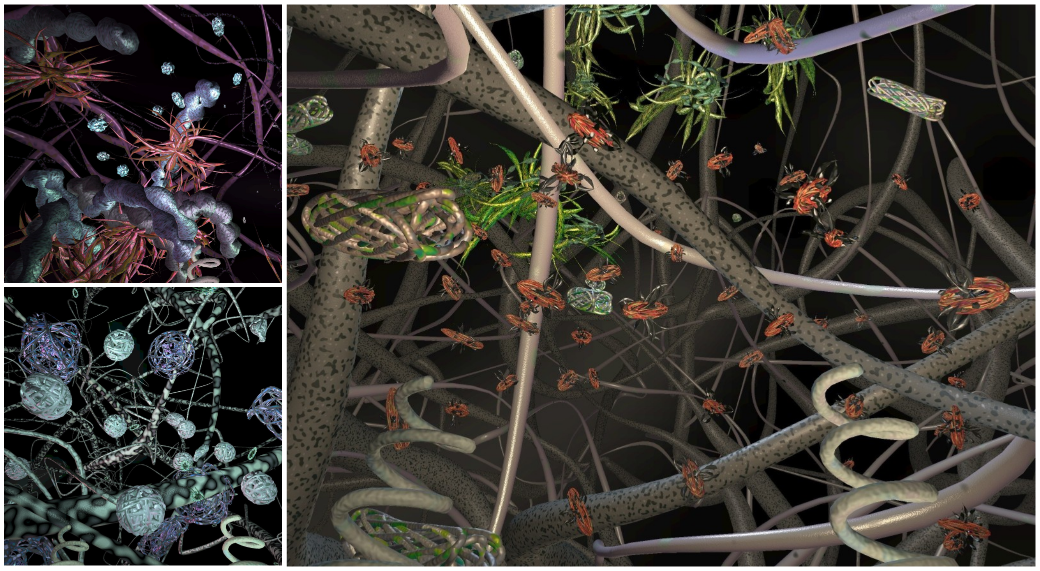
[1]https://live.newscientist.com/mutation-space-william-latham/

delivers that sort of rich and complex user experience.

We will first give some background on particle motion and use in digital and virtual art. Next, we will discuss our model of agent movement and rendering and how user interaction is implemented. Finally, we will explain additional details about how different worlds are constructed.



(a) Public engaged with the installation at New Scientist Live 2016.



(b) Screenshots of different worlds of the experience.

Figure 1: Details of *Organic Art: Vortex*.

2 Background On Particle Motion

Particle-based approaches are used extensively in both computer graphics and sound composition to define spatial trajectories of objects. One approach is to use strict mathematical relationships to define the position or velocity of particles as a function of time. One of the earliest examples of this is Whitney's differential motion [Whitney, 1980]. Here, particle velocities are configured to be integer multiples of one another to ensure "resonance", i.e., instances of synchronization, are achieved. This method is demonstrated in Whitney's composition *Arabesque* [Whitney and Cuba, 1975]. Karl Sims' particle behavior language describes several composable operations on particle states that can be used to create various effects such as vortices, spirals, and bouncing [Sims, 1990]. The animation system is demonstrated in his work *Particle Dreams* [Sims, 1988]. A similar system employing particles moving along splines is found in *Osmose* to suggest streams and swarms of insects [Davies and Harrison, 1996]. Flow fields [Reynolds, 1999] prescribe particle movements in terms of an underlying force vector field. Force systems update particle accelerations by applying Newton's second law of motion, $F = ma$, in conjunction with another law to determine the forces on particles, such as Newton's law of gravitation or Coulomb's law. Since paths are derived entirely from acceleration they have the desirable property of being C^2-smooth—a result of double integration. This leads to more natural looking motion that is largely free of sharp turns. The boids algorithm [Reynolds, 1987] is an extended force system that aims to model flocking-like behaviors of animals. It defines three rules that all particles must obey: 1) maintain a minimum distance from every other particle, 2) move in the same average direction and 3) move towards the same center of mass. Several works based on boids include *SwarmArt* [Jacob et al., 2007], *Flowspace* [Bisig et al., 2011], and *Swarm Granulator* [Blackwell and Young, 2004].

3 Agent Model

Agents are modeled as a basic particle with velocity, acceleration, and mass and keep a fixed-length history of reference frames or poses. Each pose describes a position and an orientation in the form of a tangent-normal-binormal (TNB) frame. The tangent, normal, and binormal represent the forward, up, and right vectors of the agent. TNB frames are computed solely from changes in position. This means that only an agent's path needs to be specified and its orientation will be derived automatically. In order to compute a well-behaved sequence of TNB frames over time, we found it necessary to use parallel transport [Hanson and Ma, 1995][Hanson, 2006] to minimize the amount of twisting of the frame with respect to the NB plane. A similar approach reorthogonalizes the TNB frame based on the previous normal vector, but operates under the assumption that successive differences in orientation are small [Reynolds, 1999].

4 Granular Flow Field

In mathematics, the term "flow" describes the motion of particles in a fluid or other medium. A flow field describes how particles move at each position in space. Flow at each point in space can be defined specifically as a force vector, that is, how much force is applied to accelerate a particle and in which direction.

In order to assist in automating the generation of complex patterns of motion and to allow control over the tendencies of such motions, we use a flow field throughout space. Specifically, we composed the flow field from the sum of more elementary *flow grains*, a term inspired by granular sound synthesis. A flow grain is the product of three independent components: a spherical wave, a spread function, and a direction function. The purpose of the wave portion is to permit construction of complex structure through interference patterns of multiple waves. The spread function allows the influence of the grain to vary from highly localized to spanning infinitely across space. The direction function determines the elementary pattern of flow with respect to the center of the grain. We define the flow field $\mathbf{F}$ at point $\mathbf{p}$ as a sum over N flow grains $\mathbf{F}_n$

$$\mathbf{F}(\mathbf{p}) = \sum_{n=1}^{N} \mathbf{F}_n(\mathbf{p} - \mathbf{p}_n)$$

with

$$\mathbf{F}_n(\mathbf{r}) = A_n H_n(\mathbf{r}) S_n(|\mathbf{r}|) \mathbf{D}_n(\mathbf{r})$$

where $\mathbf{p}_n$ is the position, A_n is the amplitude, and H_n, S_n, and $\mathbf{D}_n$ are the spherical, spread, and direction functions, respectively, of the n^{th} grain. To update a particle from $\mathbf{F}$, we set its acceleration vector $\mathbf{a}$ according to

$$\mathbf{a} = \mathbf{F}(\mathbf{p})/m - v_{drag}\mathbf{v}$$

where m, $\mathbf{p}$, and $\mathbf{v}$ are the mass, position, and velocity vector of the particle and v_{drag} is a drag coefficient in $[0, 1]$. Note that when $v_{drag} = 1$, we are effectively applying the force $\mathbf{F}$ to the velocity of the particle rather than its acceleration.
The following sections describe the three component functions of a flow grain in more depth.

4.1 Spherical Wave

A spherical wave, $H(\mathbf{r})$, is a periodic function in spherical coordinates that extends infinitely through space. One of the simplest types of spherical waves is what we call a *radial oscillator*. A radial oscillator is defined by

$$H^r(\mathbf{r}) = f(2\pi k|\mathbf{r}|)$$

where $f(x)$ is a periodic function satisfying $f(x) = f(x + nk)$ for integer n and k is the wavenumber or spatial frequency. For example, a radial cosine oscillator can be defined by $H^{cos}(\mathbf{r}) = \cos(2\pi k|\mathbf{r}|)$. While radial oscillators generally have a spherical shape, they can also describe planar functions in the limit. Normally, we obtain a function that is radially symmetric and hence has equal values along spherical shells. However, as $|\mathbf{r}| \to \infty$, the spherical shells flatten out into planes (Figure 2) and $\mathbf{r}/|\mathbf{r}|$ defines the direction normal to the planes.

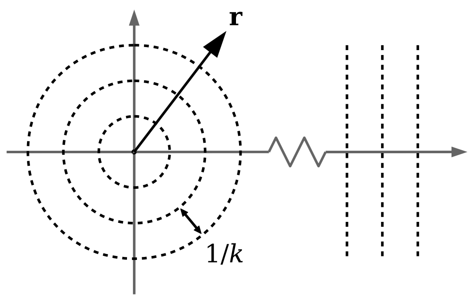


Figure 2: Illustration of a radial oscillator in 2D. The dashed lines demarcate one period, $1/k$, of the waveform. To the right, as the distance from the origin gets large, the spherical surfaces more closely approximate planes.

4.2 Spread Function

The spread function may be used to localize the influence of the grain. A spread function $S(x)$ typically satisfies

$$\begin{aligned} S(0) &= 1 \\ \lim_{|x|\to\infty} S(x) &= 0. \end{aligned}$$

where a scale mapping $x \to x/w$ can be applied to control the width w of the spread function. The identity spread function $S^u(x) = 1$ defines a global grain that has a non-vanishing effect throughout space. This can be used in situations where a flow pattern is desired throughout all space. This can give the user a stronger urge to explore space and discover different patterns of flow. To simulate the intensity falloff from a spherical radiation source, as with Newton's law of gravitation and Coulomb's law, one can use a clamped inverse-square spread function $S^{ii}(x) = 1/\max(x,1)^2$. The clamping is used to prevent the function from going to infinity as $|x| \to 0$. For a smoother falloff near $x = 0$, a Lorentzian function can be used, $S^L(x) = 1/(x^2+1)$. A useful building block is a rectangular spread function $S^{r,p}(x) = 1/(|x|^p + 1)$ where $p = 0, 1, 2, \ldots$ controls the falloff rate. For small p, $S^{r,0}(x) \equiv S^u(x)$ and $S^{r,2}(x) \equiv S^L(x)$. As $p \to \infty$, the function approaches a rectangular window. The step-like shape permits creation of highly localized regions of force or boundaries when $v_{drag} = 1$. For the user, this can create a strong association to physical objects and haptic interaction as these function through local interactions. Multiplying another function by $S^{r,p}(x)$ for large p effectively truncates it for $|x| \gtrsim 1$. This can be used to create a variety of different window functions [Harris, 1978][Nuttall, 1981]. For example a Welch (inverted parabolic) window is defined by $S^w(x) = (1-x^2) \cdot S^{r,p}(x)$. Plots of the various spread functions are shown in Figure 3.

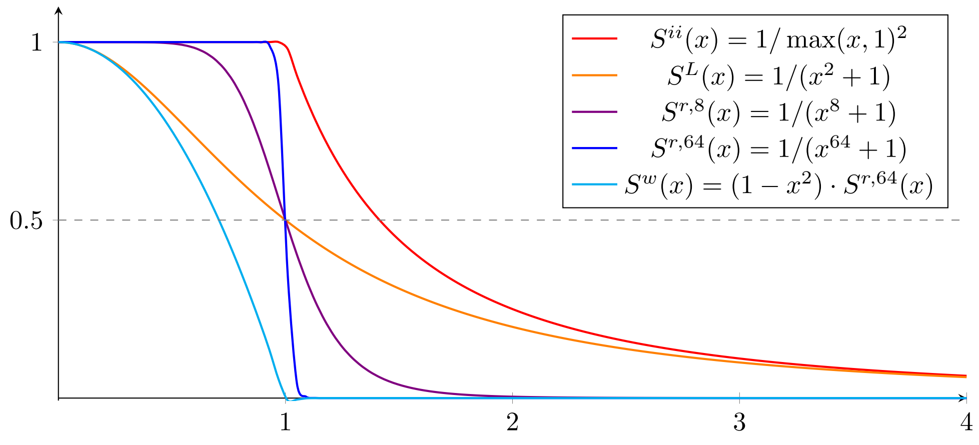


Figure 3: Examples of different spread functions.

4.3 Direction Function

The direction function determines the basic direction of flow with respect to the position of the grain. The direction function defines a field of unit vectors and can be one of two basic types: radial or angular. A radial flow moves particles outward from or inward toward a point. An outward flow corresponds to a repulsion point or source and an inward flow corresponds to an attraction point or sink. An angular flow circulates particles around an axis of rotation. The circulation can be clockwise or counter-clockwise with respect to the rotation axis. The general direction function is given by

$$\mathbf{D}(\mathbf{r}) = R(\mathbf{u}, \theta)\mathbf{r}/|\mathbf{r}|$$

where $R(\mathbf{u}, \theta)$ is a rotation matrix defining a rotation around axis $\mathbf{u}$ by angle θ. Outward radial flows have $\theta = 0$ while counter-clockwise angular flows have $\theta = \pi/2$. Inward radial and clockwise angular flows are obtained by flipping the sign on the amplitude of the flow grain.

We visualize two-dimensional radial and angular flow by using strokes representing the force as well as the divergence and curl of the field (Figure 4). Outward and inward radial flows are colored red and green, respectively, depending on the sign and magnitude of the divergence. Counter-clockwise and clockwise angular flows are indicated by magenta and cyan, respectively, and derived from the z component of the curl.

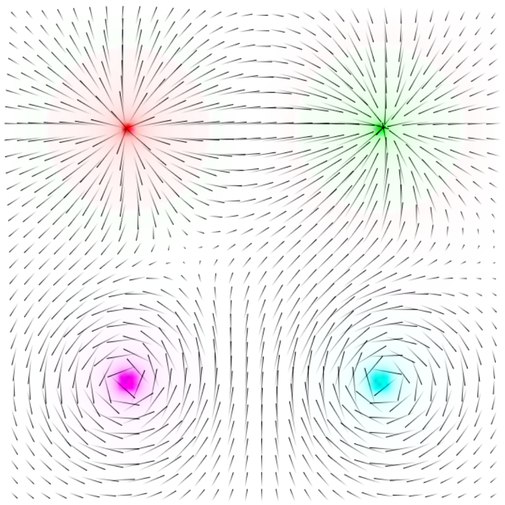


Figure 4: The basic types of flow direction. The top row shows outward and inward radial flows colored red and green, respectively. The bottom row shows counter-clockwise and clockwise angular flows indicated by magenta and cyan, respectively.

In order to achieve a stable angular flow, we found it necessary to apply a drag factor in a region around the grain. The drag factor at each point in space is a sum of contributions from each grain. Angular flow sources have a positive influence on the drag factor, while radial flow sources have a negative influence on it. The total drag factor resulting from N grains is

$$v_{drag} = \max(0, \min(1, \sum_{n=1}^{N} (\frac{4}{\pi}\theta_n - 1) A_n S_n)).$$

4.4 Flow Grain Examples

This section presents some examples of flow grains that can be used as building blocks for agent movements or user interactions. We first consider the implementation of gravitational or electrostatic sink using an inverse-square law. The flow grain is the product of an inverse-square spread function and radial direction function

$$\mathbf{F}_n(\mathbf{r}) = S^{ii}(|\mathbf{r}|)\mathbf{D}^r(\mathbf{r}) = \frac{\mathbf{r}/|\mathbf{r}|}{\max(|\mathbf{r}|, 1)^2}.$$

The resulting particle dynamic is an elliptical orbit around the center of the sink (Figure 5a).

For the second example, we consider the effect of non-vanishing spherical wave. The flow grain is given by

$$\mathbf{F}_n(\mathbf{r}) = H^{cos}(\mathbf{r})S^u(|\mathbf{r}|)\mathbf{D}^r(\mathbf{r}) = \frac{\mathbf{r}}{|\mathbf{r}|}\cos(2\pi|\mathbf{r}|).$$

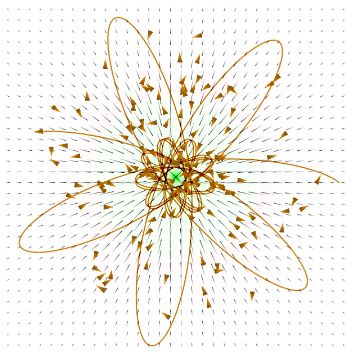
The field is comprised of alternating spherical shells of inward and outward flow. Particles oscillate around the region where the force vectors point towards one another while simultaneously circling around the center of the grain (Figure 5c). Figure 5d illustrates particle dynamics far away from the center of the field.

The third example considers a circulating flow around two points. The flow field is a sum of two flow grains with angular flow directions

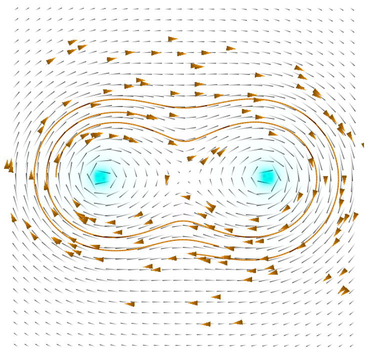
$$\mathbf{F}_n(\mathbf{r}) = S^{ii}(|\mathbf{r}|)\mathbf{D}^a(\mathbf{r}) = R_{xy}\frac{\mathbf{r}/|\mathbf{r}|}{\max(|\mathbf{r}|, 1)^2}$$

where R_{xy} is a rotation by $\pi/2$ on the xy-plane, $\mathbf{p}_1 = (-1, 0, 0)$, and $\mathbf{p}_2 = (1, 0, 0)$. The particles follow a peanut-shaped trajectory around the two flow grains (Figure 5b).

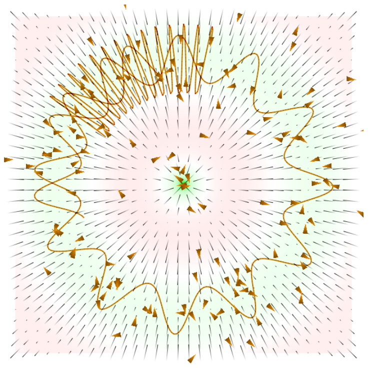
The final example is a directional sink that results in orbits that are biased in a particular direction (Figure 5e). It is composed of an sink and source that are positioned near one another. The source generally has a narrower spread and lower amplitude than the sink so as to keep the combined field attractive. The direction of the bias points from the source to the sink.



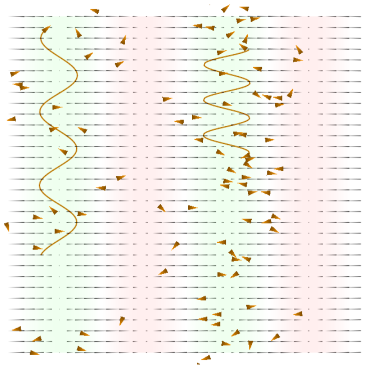
(a) Gravitational sink



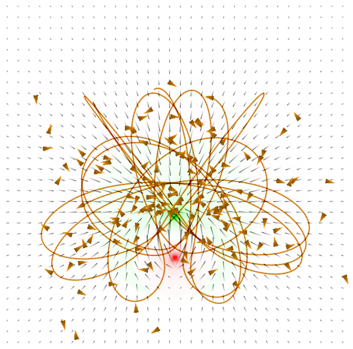
(b) Two-point circulation



(c) Oscillating spherical field



(d) Oscillating planar field



(e) Directional sink

Figure 5: Examples of various 2D flow fields. Line strokes show the direction and magnitude of the force field. Triangles represent agents and the curves show the trajectories of two agents. Green and red areas show the divergence of the field and indicate regions of attraction and repulsion, respectively.

5 Implementation of Interactions

The primary goal for interaction was to give users a sense of play with autonomous agents. Besides the headset, the primary form of control are two HTC Vive controllers, one for each hand. The controllers are represented in virtual space by two coils that track the movement of the controllers.

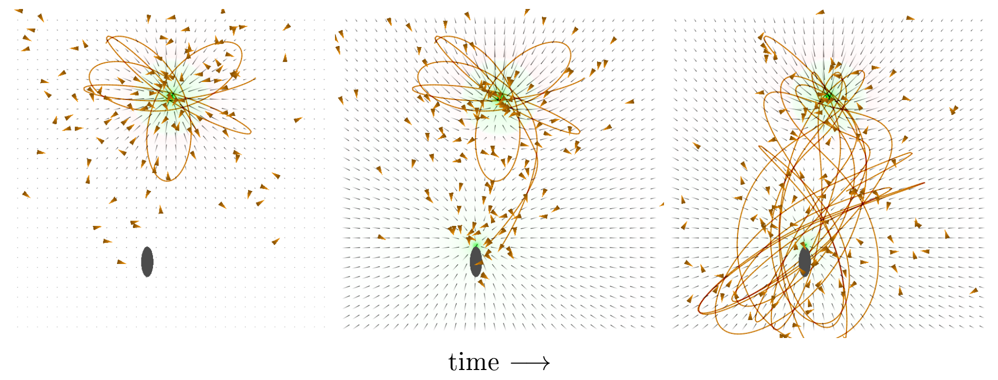
5.1 Navigation

Although the headset provides a three-dimensional position and orientation in a 4.6 m^2 tracking space, we implemented a secondary mode of navigation to allow the user to move more freely about the space. Using a thumb-operated two-dimensional analog trackpad on the top of the controller, the user can move in the direction that the controller is pointing. For example, if the user pushes up on the trackpad, they will move along the forward direction vector of the controller and if they push right on the trackpad, they will move along the right direction vector of the controller. We only allow the user to translate with the controller as we found that performing any kind of rotation of the user's view not originating from head tracking quickly led to an uneasy feeling.

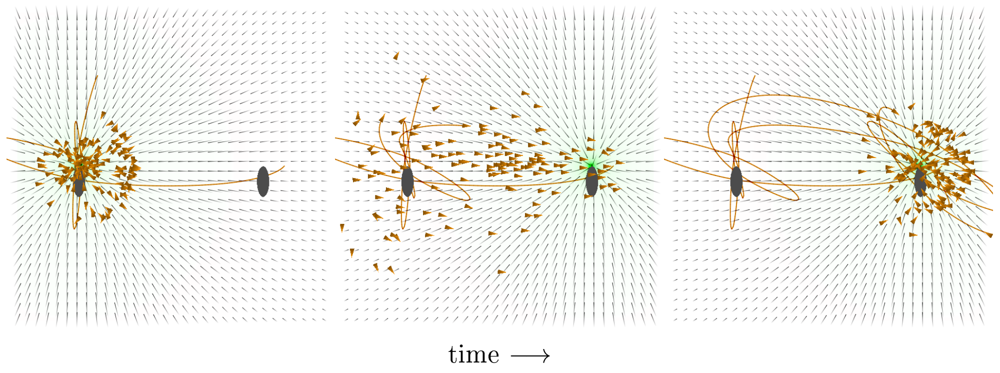
5.2 Agent Interaction

Agent interactions are accomplished through use of various flow grains. There is a highly-localized repulsion point at the controller tip that is always active. This allows agents to be pushed away with the controllers. By pressing the analog trigger located on the bottom of the controller, the user is able to control the strength of a flow grain attached to the tip of the controller. When the controller trigger is pressed an attraction point is mixed in to the flow field to pull the agents towards the user. This attraction point has a lower amplitude and higher spread than the repulsion point so that it can attract agents far away and not diminish the effect of the repulsion point. In addition, a small amount of drag is added to the attraction point to prevent the agents from oscillating around it indefinitely and going out of the user's view. Furthermore, the drag factor adds a strong sense of autonomy to the agents as if they decide to slow down when near a point of interest. When the agents oscillate around the attraction point (without drag), they appear to be less in control of their movement.

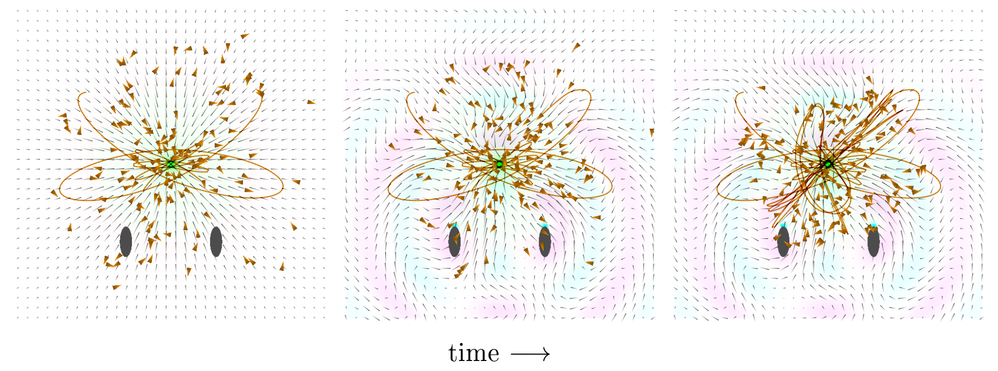
We consider three scenarios of user interaction with agents. For simplicity, we restrict user interaction to turning on or off two fixed attraction points. In practice, the user can control the strength and position of each force. In the first scenario, a user creates an attractive field to draw agents towards the tip of one of the controllers. Figure 6a shows how the interaction progresses over time with time going left to right. In the first image, the agents orbit an attraction point without interaction. The middle image show the moment just after the user creates an attractive force. Here, we see the agents in transit towards the new force and the user experiences a basic sense of connection with the agents. The final image displays how the agents behave after a prolonged period of time. Even in such a simple scenario, agent behaviors can be complex and unpredictable. While most agents approach and remain in the vicinity of the user, some may never approach the user, and others may exhibit more sporadic behavior between the former extremes. In the second scenario, the user creates a stream of agents by controlling two attraction points. Figure 6b illustrates the process, again with time moving left to right. First, the user activates an attraction point on the left and holds it steady for some amount of time in order to create a cluster of agents. After some time, the left attraction point is turned off and a second one is turned on on the right. This causes the cluster to break up and form a stream moving from left to right. The process can be repeated to "juggle" the agents back and forth. In the final scenario, the user has the ability to create complex flow patterns from interference of the two controllers (Figure 6c). Each controller has a radially-oscillating angular flow that when mixed create a complex flow of vortex-like motions. As opposed to the other scenarios, this type of interaction acts more to swirl the agents rather than attract them.



(a) Attracting agents away from a sink.



(b) Creating a stream of agents between two sinks.



(c) Interrupting a simple sink with a complex angular flow pattern.

Figure 6: Illustrations of user interactions with agents. The dark gray ovals are the user controllers, the triangles are agents, and the curves are the paths of two randomly selected agents.

On the head of the user, tracking the position of the HMD, we place a repulsion point with a spread matching the dimensions of a human head. The main purpose of this is to keep agents from getting too close to the user's view and therefore obstructing it. We also found it increased the immersant's sense of the agents being aware of their presence in the space. Along similar lines, we experimented with adding an attraction point following the forward vector of the user's head. The idea was to allow the agents to persist in the user's field of view even without interaction to add a further sense of agents' awareness of the user. However, we abandoned this idea as we found that it tended to clutter the view of the user and diminish the sense of depth of the scene.

6 Curve-based Synthesis

Curve-based synthesis employs space curves as a unified substrate for simultaneous generation of shapes and sounds. The curves are generated from sums of harmonic spherical curves [Putnam, 2014b] which are capable of producing a wide array of shapes and textures with a relatively small set of parameters. The curves generated in this manner have several additional advantages

- closed and sequential and thus translate directly into sound waveforms
- allow precise control over symmetry/asymmetry
- describe many basic 3D shapes such as spheres, cylinders, hyperboloids and tori
- trivial to parallelize computation

Implicit surface generators and L-systems may be more general systems, however, the resulting geometry is not one-dimensional and thus cannot be directly translated into sound waveforms. Another potential complication with these systems is how to smoothly interpolate from one geometric structure to another.

Visually, the space curves are rendered as textured tubes and sonically, they are used as a set of sound-producing waveforms. For each world, the curve parameters are determined from random distributions along with fixed constraints to ensure variation while maintaining specific properties of the curve. To obtain more geometric complexity over a simple tube, we apply a "ribbing" effect [Todd and Latham, 1992] that modulates the radius of the tube along its length. These space curves are used to generate the graphics and sounds of two elements of the work—the agents and the environment.

6.1 Agent Geometry

The bodies of the agents consist of two superimposed space curves: one for the main body and one for filaments extending from the body. The curve generation algorithms are constrained to produce shapes with either bilateral or radial symmetry in order to create more convincing organisms resembling those in broad animal ranks such as bilateria or cnidaria. The filaments suggest various types of appendages such as antennae, tentacles, barbs, spines, wings, and arms or microtubular structures such as cilia, flagella, and spindle apparatuses.

For the body curves, we employ the space curve function

$$A\{a\}_Z\{b\}_Y\{0,\theta\}_X\{c\}_Y + (1-A)\{a\}_Z\{f\}_Y\{0,\phi\}_X\{g\}_Y. \tag{1}$$

Mathematical details of (1) can be found in the appendix. The function produces a wide array of curves lying on spheres, spherical caps and zones, and other more complex surfaces of revolution. This function has the desirable characteristic of providing exact control over the order of dihedral symmetry of the curve. Specifically, the resulting space curve exhibits m^{th} order dihedral symmetry with respect to the z axis when $m = \gcd(b, c, f, g)$ and a and m are coprime. Thus, to generate a "random" curve with m^{th} order dihedral symmetry, first choose a coprime with m, then b, c, f, g as integer multiples of m, and finally, random values of A, θ, and ϕ.

The filament curve is constrained to self intersect along the polar axis to give a preliminary branching structure. Conveniently, this can be accomplished by setting

$\theta = 0, \phi = 1/2$ in (1). Curve sections beyond a certain radius are then cut away to produce the appearance of individual filaments. Cutting is accomplished simply by shrinking the radius of the tube down to zero. Figure 7 illustrates the filament cutting operation on a 2D cross section. In order to obtain fluid body movement, the curves are mapped onto a fixed history of the agent's TNB frames. The curve z coordinate is mapped to time to retrieve a TNB frame and then the curve x, y coordinates are rotated onto the NB plane. Finally, for both curves, the number of ribs is set to an integer multiple of one of the frequencies used to generate the curve in order to maintain symmetry. Figure 8 shows a selection of body shapes produced from the methods described.

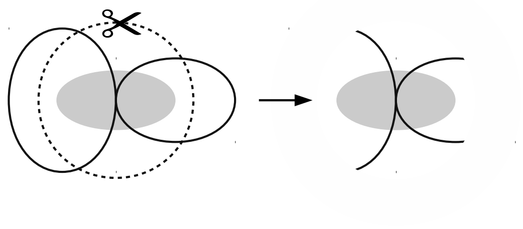


Figure 7: Illustration of filament cutting operation. On the left is the original (closed) filament curve shown in black. The dashed circle shows the cutting radius beyond which the filament curve is cut away. On the right is the result of the cutting operation—a branching structure. The gray ellipse represents the body curve for reference.

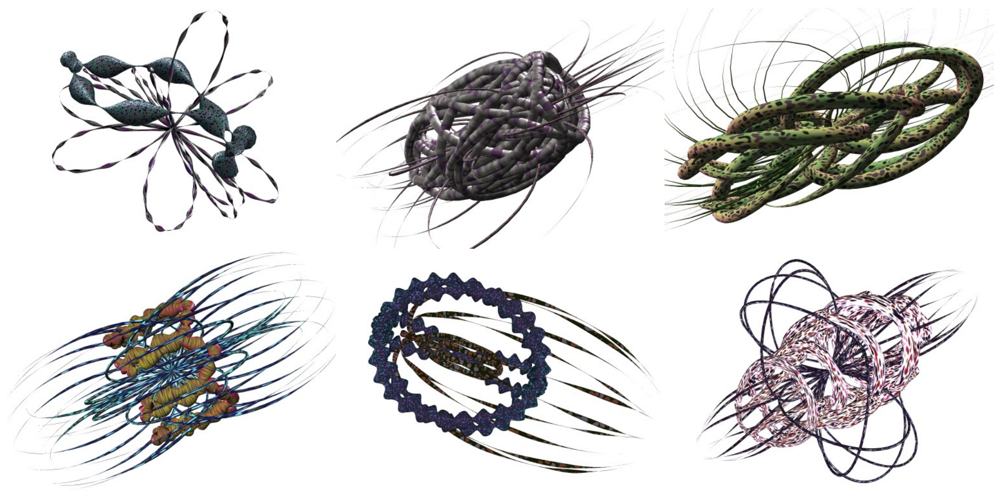


Figure 8: A selection of agent bodies generated from space curves.

6.2 Agent Voices

Each agent has a unique "voice" that is derived directly from its body shape. Agent voices are localized according to distance cues for azimuth and parametric filters derived from pinna spectral notch data [Raykar et al., 2005][Iida et al., 2007] for elevation. A voice consists of three independent waveforms that get combined to generate the final sound: a tone, a portato, and a chirp. The sound is constructed by modulating the frequency of the tone by the chirp and then multiplying the resulting tone by the portato. In general, the portato and chirp waveforms are set several octaves below the tone frequency. The portato waveform requires a special mapping as explained below. The three waveforms originate directly from the space curve and are converted into sound using wavetable synthesis. Since the space curves are three-dimensional, a single wavetable is generally the x, y, or z component of one such curve. For more variation, linear mixtures of the x, y, and z components are used. To ensure the portato waveform has a more impulsive shape, we apply a special waveshaping function after the table lookup that produces impulse-like shapes at zero crossings and approximately zero elsewhere. To accomplish this, we use a normalized Lorentzian function (figure 9), a singly peaked function centered at $x = 0$. It is given by

$$L(x, c) = \frac{c^2}{c^2 + x^2}$$

where c is the peak half-width so that $L(c, c) = 1/2$. Small values of c produce narrow peaks leading to patterns of clicks, pops, chirrups, and other granular effects. Increasing c results in more tonal, tremulant sounds. The pre-waveshaped portato waveform should have an adequate number of zero crossings over one period to ensure an interesting pattern of impulses. In general, we found it helpful to low-pass filter the portato waveform to diminish highly-impulsive portions of the waveform.

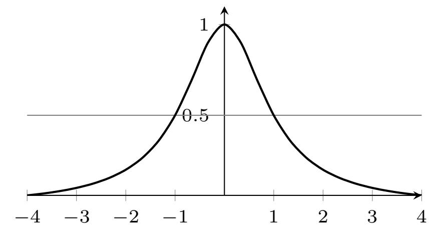


Figure 9: Lorentzian function $L(x, c) = \frac{c^2}{c^2+x^2}$ for $c = 1$.

For this work, we use the body curve for the tone and the filament curve for the portato and chirp. The frequency of the chirp is set four octaves above that of the portato in order to help ensure the frequency modulation is heard. In order to provide some variation in the voicing of the agent, we map its velocity onto an interpolation fraction that mixes the curves' x, y, and z components into the respective voice waveform. This direct sonic effect helps enhance many of the basic interactions with the agents involving attraction or repulsion.

6.3 Environment Generation

The environment consists of vine-like structures that permeate the space and provide ambient background. The environment consists of a primary "trunk" curve and a secondary curve to ornament the trunk. The two curve shapes are identical except for an additional high-frequency, low-amplitude component added to the secondary curve. This causes the secondary curve to oscillate around the primary curve in a variety of patterns. The vine is translated into sound via vector synthesis [Roads, 1996] of the individual x, y, and z components of the curve. To ensure motion in the sound, mixing is automated through cyclic interpolation between pairs of components over a total period of 45 seconds (Figure 10).

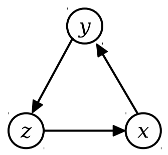


Figure 10: Vector synthesis scanning pattern used to translate a space curve x, y, and z components into a sound.

7 World Generation

7.1 Components

Each world consists of a fixed number of agents, a fixed number of flow grains, and environmental vines. The agents are divided into three genera where each genus has a specific mass and body shape that the individual species inherit with small random modifications. The size of each agent is made proportional to its mass to add a degree of physical realism. Additionally, the number of agents in any given genus is made inversely proportional to the mass/size prescribed by the genus. This proportioning is done to help reduce the amount of graphics rendering and mitigate potential occlusion from larger agents. Flow grains are placed in random positions for each world and are slowly amplitude modulated over time to help prevent agents from getting into static orbits or equilibrium states when there is no interaction from the user. The principle is the same as the streaming interaction described in section 5.2.

7.2 Morphing

A world morphing technique was employed to allow a seamless transition from one environment to another. Morphing is mostly a trivial operation only requiring interpolation of synthesis parameters. However, interpolation of some parameters leads to undesirable effects. One problematic set of parameters are the frequencies used to generate the space curves. These curves rely on the frequencies being harmonically-related to ensure the curves are closed, that is, made of a single loop. The closed curves are especially important for tone production since any discontinuities in the waveform will lead to harmonic distortion. To solve this problem, we store the space curves in tables and then interpolate between the tables.

8 Technical Detail

Mutator VR: Vortex uses the HTC Vive[2] virtual reality headset which includes room-scale tracking, two wireless hand-held controllers, and two channels of audio. Graphics rendering uses an Nvidia GeForce GTX 1080. Custom software is implemented in OpenGL/GLSL and C++ using the AlloSystem virtual environment toolkit[3], Gamma sound synthesis library [Putnam, 2014a], and OpenVR SDK[4].

9 Conclusion

The goal of *Mutator VR: Vortex* is to provide a continuum of rich and unexpected interactive multisensory experiences in virtual reality. Accomplishing this requires selection of suitable mathematical constructs that can provide a natural integration of modalities. Overall, we found curved-based synthesis and flow grains to provide this kind of integration. Closed space curves generated from sums of spherical curves

[2]https://www.vive.com
[3]https://github.com/AlloSphere-Research-Group/AlloSystem
[4]https://github.com/ValveSoftware/openvr

produce a large variety of shapes and timbres and are straightforward to translate to graphics and sound. Having precise control over the symmetries and path of the curves was helpful in automating generation of organic-looking forms. Flow grains enabled complex patterns of motions to be automatically generated and lent themselves towards emergent types of interaction such as juggling agents from one controller to the other or swirling agents into a vortex. Even with very simple flow grains, we found it possible to create a convincing sense of agent autonomy and playful and emergent interactions. We relied heavily on random distributions for choosing parameters for each world which certainly led to unexpected results. However, there is an underlying sense of lack of unity in many of the worlds which may be due to the random elements. It is possible that this could be addressed through more structured parameter selection or a deeper construct to tie all pieces of each world together. The latter may be a more fruitful direction to explore as we have obtained good results using the integrated methods described here.

10 Acknowledgments

This work was conducted in the Digital Creativity Labs (www.digitalcreativity.ac.uk) funded by the EPSRC, AHRC and InnovateUK under grant number EP/M023265/1. We also thank Frederic Fol Leymarie for his feedback on early drafts.

Appendix

The divergence of a vector field indicates the amount of outward radial flow from a given point. For a given vector field $\mathbf{F}$, its divergence is the scalar

$$\nabla \cdot \mathbf{F} = \frac{\partial \mathbf{F}_x}{\partial x} + \frac{\partial \mathbf{F}_y}{\partial y} + \frac{\partial \mathbf{F}_z}{\partial z}.$$

The curl of a vector field measures the amount and direction of infinitesimal rotation at a given point and therefore is an indication of angular flow. It is given by the vector

$$\nabla \times \mathbf{F} = \left(\frac{\partial \mathbf{F}_z}{\partial y} - \frac{\partial \mathbf{F}_y}{\partial z}, \frac{\partial \mathbf{F}_x}{\partial z} - \frac{\partial \mathbf{F}_z}{\partial x}, \frac{\partial \mathbf{F}_y}{\partial x} - \frac{\partial \mathbf{F}_x}{\partial y} \right)$$

where $|\nabla \times \mathbf{F}|$ is the amount of rotation and $\nabla \times \mathbf{F}/|\nabla \times \mathbf{F}|$ is the axis of rotation.
A spherical curve of N points is generated by rotating the vector $(0, 0, 1)$ by the matrix

$$\alpha_n = A\{f_1, \theta_1\}_Z \{f_2, \theta_2\}_Y \{f_3, \theta_3\}_X \{f_4, \theta_4\}_Y$$

where n is the n^{th} point along the curve, A is the amplitude, and $\{f, \theta\}_e$ is a rotation matrix describing a rotation around axis e by the angle $2\pi(nf + \theta)$ radians [Putnam, 2014b]. For brevity, we define $\{f\}_e \equiv \{f, 0\}_e$.

References

[Bisig et al., 2011] Bisig, D., Schacher, J., and Neukom, M. (2011). Flowspace—a hybrid ecosystem. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 260–263, Oslo, Norway.

[Blackwell and Young, 2004] Blackwell, T. and Young, M. (2004). *Applications of Evolutionary Computing: EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoISAP, EvoMUSART, and EvoSTOC, Coimbra, Portugal, April 5-7, 2004, Proceedings*, chapter Swarm Granulator, pages 399–408. Springer, Berlin, Heidelberg.

[Davies and Harrison, 1996] Davies, C. and Harrison, J. (1996). *Osmose*: Towards broadening the aesthetics of virtual reality. *Computer Graphics (ACM SIGGRAPH)*, 30(4):25–28.

[Gilbert, 2011] Gilbert, R. L. (2011). The P.R.O.S.E. (Psychological Research on Synthetic Environments) Project: Conducting in-world psychological reserach on 3d virtual worlds. *Journal of Virtual Worlds Research*, 4(1):3–18.

[Hanson, 2006] Hanson, A. J. (2006). *Visualizing Quaternions*. Morgan Kaufmann Publishers, San Francisco, CA.

[Hanson and Ma, 1995] Hanson, A. J. and Ma, H. (1995). Parallel transport approach to curve framing. Technical report.

[Harris, 1978] Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83.

[Iida et al., 2007] Iida, K., Itoh, M., Itagaki, A., and Morimoto, M. (2007). Median plane localization using a parametric model of the head-related transfer function based on spectral cues. *Applied Acoustics*, 68:835–850.

[Jacob et al., 2007] Jacob, C. J., Hushlack, G., Boyd, J. E., Nuytten, P., Sayles, M., and Pilat, M. (2007). SwarmArt: Interactive art from swarm intelligence. *Leonardo*, 40(3):249–254.

[Lewis, 2008] Lewis, M. (2008). *The Art of Artificial Evolution*, chapter Evolutionary Visual Art and Design, pages 3–37. Springer, Berlin.

[Nuttall, 1981] Nuttall, A. H. (1981). Some windows with very good side-lobe behavior. *IEEE Transactions on Acoustics Speech and Signal Processing*, 39(1):84–91.

[Putnam, 2014a] Putnam, L. (2014a). Gamma: A C++ sound synthesis library further abstracting the unit generator. In *Proceedings of the International Computer Music Conference and Sound and Music Computing 2014*, pages 1382–1387, Athens, Greece.

[Putnam, 2014b] Putnam, L. (2014b). A method of timbre-shape synthesis based on summation of spherical curves. In *Proceedings of the International Computer Music Conference and Sound and Music Computing 2014*, pages 1332–1337, Athens, Greece.

[Raykar et al., 2005] Raykar, V. C., Duraiswami, R., and Yegnanarayana, B. (2005). Extracting the frequencies of the pinna spectral notches in measured head related impulse responses. *Journal of the Acoustical Society of America*, 118(1):364–374.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

[Reynolds, 1999] Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference 1999*, pages 763–782, San Jose, CA.

[Roads, 1996] Roads, C. (1996). *The Computer Music Tutorial.* MIT Press.

[Sims, 1988] Sims, K. (1988). Particle dreams.

[Sims, 1990] Sims, K. (1990). Particle animation and rendering using data parallel computation. *Computer Graphics*, 24(4):405–413.

[Todd and Latham, 1992] Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers.* Academic Press, London.

[Whitney, 1980] Whitney, J. (1980). *Digital Harmony: On the Complementarity of Music and Visual Art.* Byte Books, Peterborough, NH.

[Whitney and Cuba, 1975] Whitney, J. and Cuba, L. (1975). Arabesque.

List of Figures