social sqncr

An interactive audio-visual installation

By Nick Hardeman and Bruce Drummond

Thesis Instructor: Marko Tandefelt
Thesis Writing Instructor: Loretta J. Wolozin

A thesis document
submitted in partial fulfillment
of the requirements for the degree of
Master of Fine Arts in Design and Technology
Parsons The New School for Design
May 2010

Acknowledgements

We would like to thank our families for their love and support. Zachary Lieberman for his tutelage and guidance. To the Open Source community for all their help and suggestions. And most of all, to Sava and Victoria who stuck it out with us through many sleepless nights.

Abstract

social sqncr is an interactive audio-visual installation that aims to make people more aware of the act of creating their public identity.

To create their 'identity', participants use their bodies to physically interact with a pseudo-musical instrument projected in a physical space to create virtual musical creatures based on users' movements. The instrument consists of eight zones that participants interact with which influence the shape, movement, and aural properties of the resulting entity.

Once complete, the system captures an image of the participant to attach to the creature which is then set free into an eco-system inhabited by similar entities. The entities react to one another, much like human beings in a social network. Participants can cause environmental disturbances by physically interacting with the projection.

The physical movements required of participants ensures a high level of investment in the process, causing a heightened awareness of their creation, and thus themselves. Watching their entity interact with other entities is a reflection of their own interaction in their social networks.

Table of Contents

List of Illustrations

Chapter I. Introduction

1. Concept

social sqncr is an interactive audio-visual installation that aims to make people more aware of the act of creating their public identity.

To create their 'identity', participants use their bodies to physically interact with a pseudo-musical instrument projected in a physical space to create virtual musical creatures based on users' movements. The instrument consists of eight zones that participants interact with which influence the shape, movement, and aural properties of the resulting entity.

Once complete, the system captures an image of the participant to attach to the creature which is then set free into an eco-system inhabited by similar entities. The entities react to one another, much like human beings in a social network. Participants can cause environmental disturbances by physically interacting with the projection.

The physical movements required of participants ensures a high level of investment in the process, causing a heightened awareness of their creation, and thus themselves. Watching their entity interact with other entities is a reflection of their own interaction in their social networks.

2. Audience and Setting

This project was developed for an indoor gallery setting and can work under varying lighting conditions. We designed and programmed the project to be set up in a space that has controlled spatial and lighting constraints, but it can be installed outdoors and/or in larger spaces by making a few minor adjustments and tweaks to the settings.

The audience for the project is varied in terms of demographics and social types. We specifically want to target social network users, social gamers and casual gamers for purposes of this project, but there is appeal for a wider audience as well. One of the objectives of the project was to encourage multi-user collaboration and interaction, while still providing an engaging single user experience.

3. Impetus

The impetus for this project stems from various areas of interest.

Algorithmic art and music, virtual living systems

For this project we wanted to explore the area of algorithmic art to create a highly ordered system that included algorithmically generated creatures that exhibited behaviors based on a predefined set of rules. This also tied directly into our interest of creating a virtual living system.

Interactive audio-visual experience

We wanted to build an engaging audio-visual experience with the ability for users to participate, interact, and alter the behavior of the creatures within the system. Each creature can be thought of as a music sequencer that participants create in real-time. They can also interact with existing creatures to modify their aural properties.

Projection

Visual presentation of the system was very importance to us. We explored the possibilities of projecting the system on a large surface using an overhead projector to increase the immersive nature of the project and engage the participants further.

Non-traditional interface

One of our goals was to move away from traditional interfaces like screen-based media that used traditional inputs like mouse and keyboards, that people were used to. This led us to a gestural input system in which participants use their bodies as the input mechanism.

Encourage participatory, collaborative, and performative behavior
Not only did we want to create an alternative interface that would be engaging, we wanted people to have fun, be social and collaborative, and exhibit performative behavior while interacting with our project. This was a challenging design question that we needed to address.

4. Design Questions

- How can this installation be inviting and intuitive to a wide spectrum of users?
- How can we make the system robust enough to account for changes in environmental light in the installation space?
- How can we create an emotional connection between the user and their virtually created entity?
- How can we capture users' experience in a public space and display it in an interesting way?
- How can we make the interaction fun and engaging?

Chapter II. Domains and Precedents

1. Domains

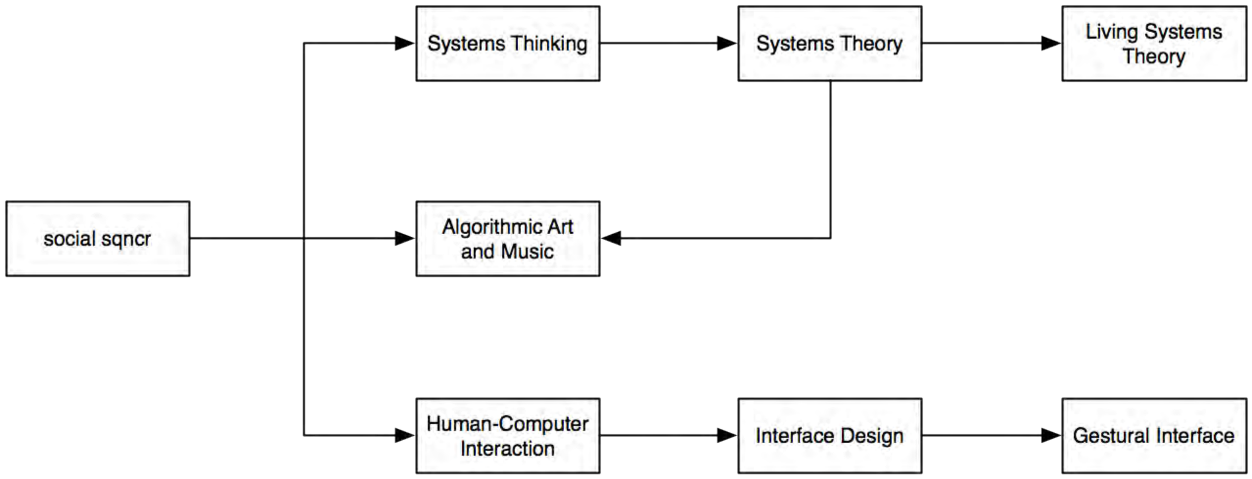


Figure 2.1 - Domain Map

a. Systems Thinking

social sqncr was envisioned as an abstract representation of a system which is a sum of its components. Each of the components has its own qualities and bears a direct relationship to other components. This idea deals with system dynamics and falls under the systems thinking domain, specifically systems theory which investigates how any group of objects works together to produce certain results within complex systems, in nature, society and science.

Another subset of the systems thinking domain that the project falls under is the living systems theory created by James Grier Miller [8]. This is a general theory about the existence of all living systems, their structure, behavior, interaction and development.

b. Algorithmic Art and Music

Algorithmic art is a subset of "... generative art that refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art." [2]
The components of the system - the creatures - are constructed and generated in an algorithmic manner to achieve autonomous behavior and visual characteristics. Human interaction alters the algorithm to produce varied results in real-time.

Each component/creature has a characteristic tone and is made up of a dynamic series of algorithmically generated musical notes and scales. The tempo of the creatures also varies based on parameters calculated during the user interaction. The sound is structured in a way to generate patterns within a defined range of scales and musical notes. The sound is also used to indicate disturbances caused by human interaction, in the event of which, it causes a change in the timbre of the sound of the effected creature.

c. Human-Computer Interaction

The project uses an intuitive gestural interface that participants interact with using physical body movements to create their own creatures via a pseudo-musical instrument and can also affect creatures in the virtual environment.

2. Precedents

a. **Terrarium - Theo Watson and Emily Gobeille (2008) [23]**

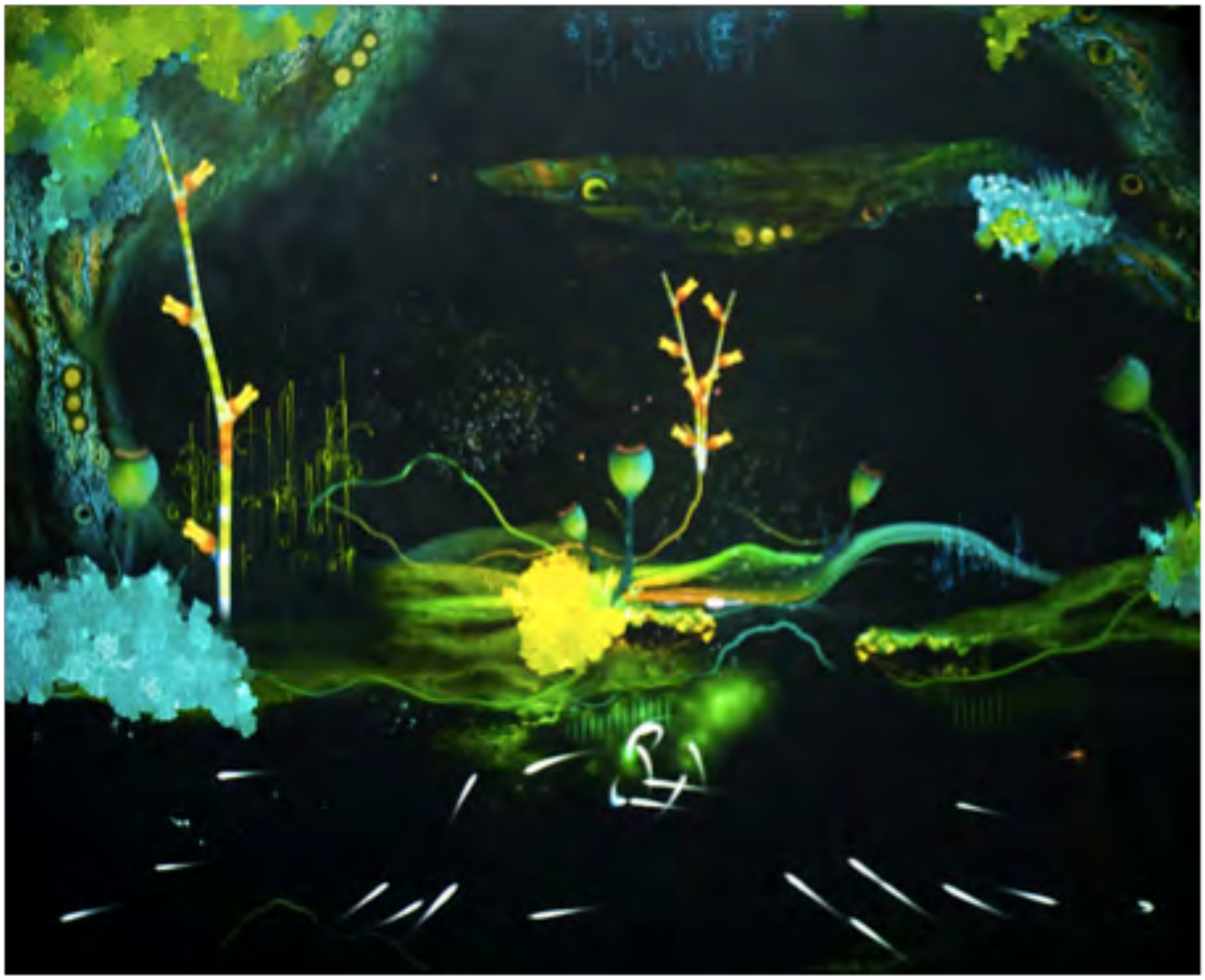


Figure 2.2 - Terrarium by Theodore Watson and Emily Gobeille.

"Terrarium is an interactive, sonic ecosystem whose source of energy comes from the sounds people make interacting with the work. The sounds of the participants voices come into the world via the sound vents." [23]

We studied several aspects of Terrarium that are related to our project and have been executed well. Terrarium is also a generative environment in the form of an ecosystem which is manipulated by user input. However, the user's manipulation is controlled by audio and it does not include visual input. The visuals of the ecosystem reacts to and evolve based on audio input from users and the surrounding physical environment. We also are drawn to the detailed visual style. The graphics work well individually and as a composition. The larger graphics maintain their position, while the smaller ones frolic in the environment and helps maintain a nice, consistent composition while still having many moving "parts."

b. Pi@Glastonbury - Memo Akten (2008) [1]

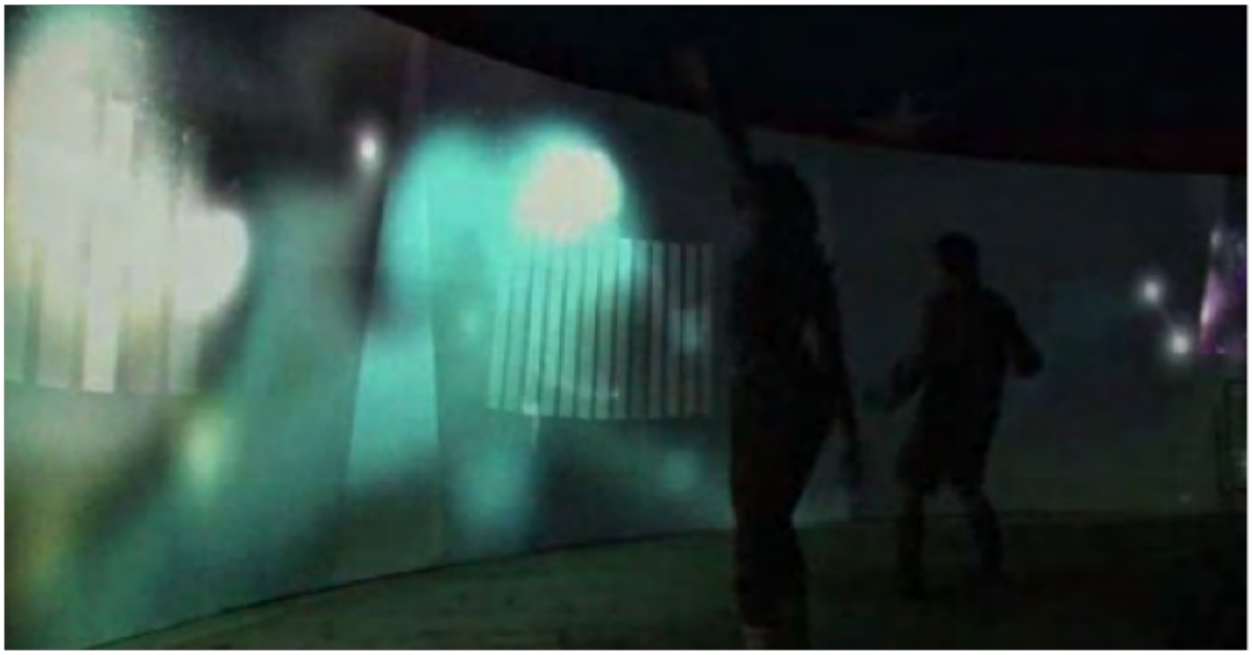


Figure 2.3 - Pi@Glastonbury by Memo Akten.

"Pi" is an audio/visual installation that allows users to trigger "zones" on various screens that create tones. The six interactive screens promote collaboration as "balls of light" travel back and forth between participants.

This audio / visual installation has been a great inspiration to our thesis because of the input methods utilized. This project uses body movement as input. The use of the user's silhouette to trigger tones promotes users to move their body and the audio generated from their movement encourages them to dance. We appreciate how much the users are compelled to move in unique ways. We also admire the compelling visuals. The fluid-like visuals are intriguing, dynamic and visually stunning. The installation of this project was large and intriguing. The six large interactive screens are arranged in a half-moon and offers a welcoming space to participants, as well as, enough space to move and dance to interact with the piece. We also favor the music composition. All of the sounds generated by the users are played at once, and relies on the users to compose the music. This project also promotes a collaboration. "Balls of light" travel between screens which trigger sounds, so one user can pass the ball to another user, along with all user's audio being played is a great collaborative experience.

c. Video Place - Myron Krueger (1975) [7]

Figure 2.4 - Still from Video Place by Myron Krueger.

"There are about 25 different pre-programmed Interaction patterns for one or more participants; in single participant mode Videoplace is more meditative in character, in multi-participant mode more playful. In its first stage in 1975, Videoplace allowed two players in remote locations to interact with with each over a combined video projection.Krueger later also developed artificial or human 'critters' that interact with the silhouette of the participant." [7]

We must mention Videoplace as a precedent for our work because Myron Krueger was a pioneer in computer vision. He demonstrates several different types of interaction that are simple and clever. Several scenes include users moving physically to interact with virtual entities, which influenced our interaction model, specifically with user's interaction with the creatures.

d. Audiovisual Environment Suite – Golan Levin (1998 – 2000) [8]

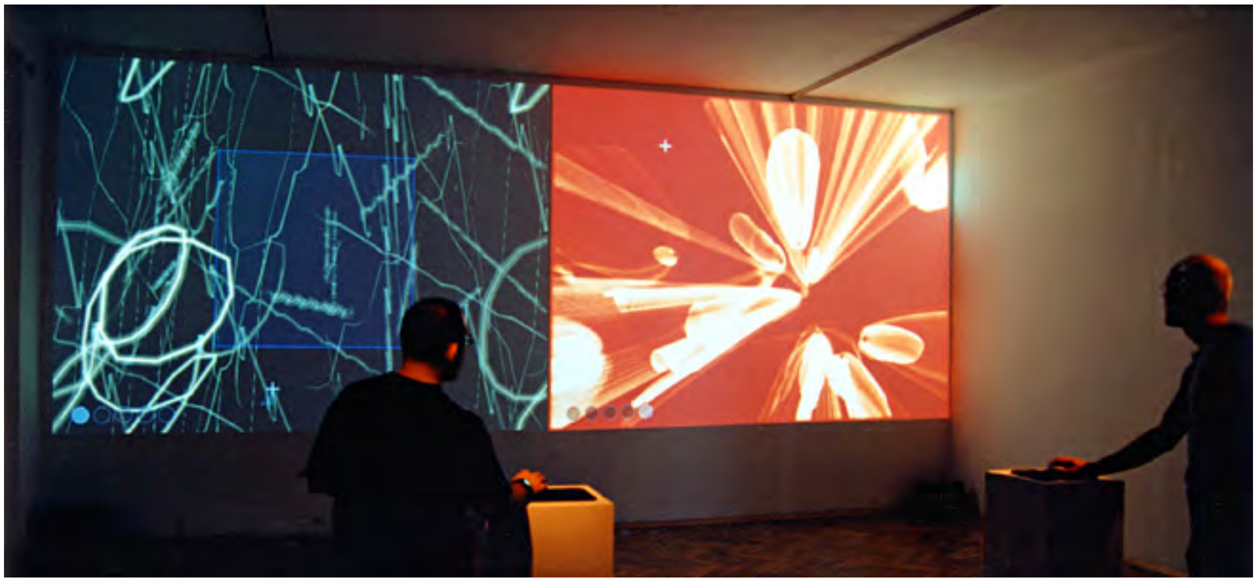


Figure 2.5 - Audiovisual Environment Suite by Golan Levin.

"Audiovisual Environment Suite is set of seven interactive software systems which allow people to gesturally create and perform abstract animation and synthetic sound in real time. Each environment is an experimental attempt to design an interface which is supple and easy to learn, yet can also yield interesting, infinitely variable and personally expressive performances in both the visual and aural domains." [8]

"The AVES systems are based on the metaphor of an 'audiovisual substance' which can be gesturally created, manipulated, and deleted in a painterly, non-diagrammatic image space." [8]

The gestural input, aesthetic qualities and ease of interaction of this project are what inspired us. We also liked the tight relationship between the dynamic visuals and computer generated sound. The project encourages the user to explore the interaction area and discover the results in real time.

e. The Grand Piano at FAO Schwarz [3]

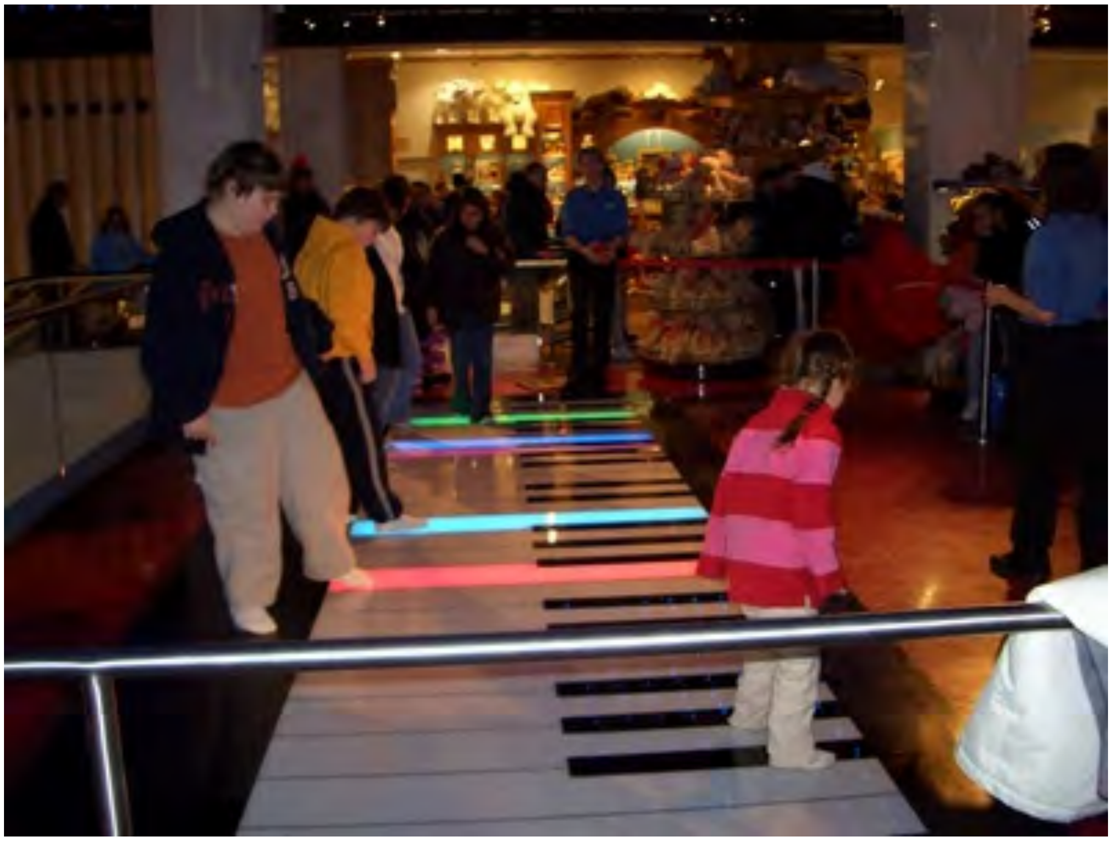


Figure 2.6 - The Grand Piano installed at FAO Schwarz on fifth avenue in New York City.

This large scale floor piano is installed at the FAO Schwarz flagship store located on fifth avenue in New York City. Users simply apply pressure to each key, resulting in a highlighted color and corresponding tone. The interaction is simple and requires large physical movements to create interesting rhythms and beats. We also like the large format because it allows for collaborative music making between several users. Tom Hanks and Robert Loggia danced "Heart & Soul" and "Chopsticks" in the 1988 feature film Big. [3]

f. GENMA – Genetic Manipulator – Christa Sommerer and Laurent Mignonneau (1996 – 1997) [19]

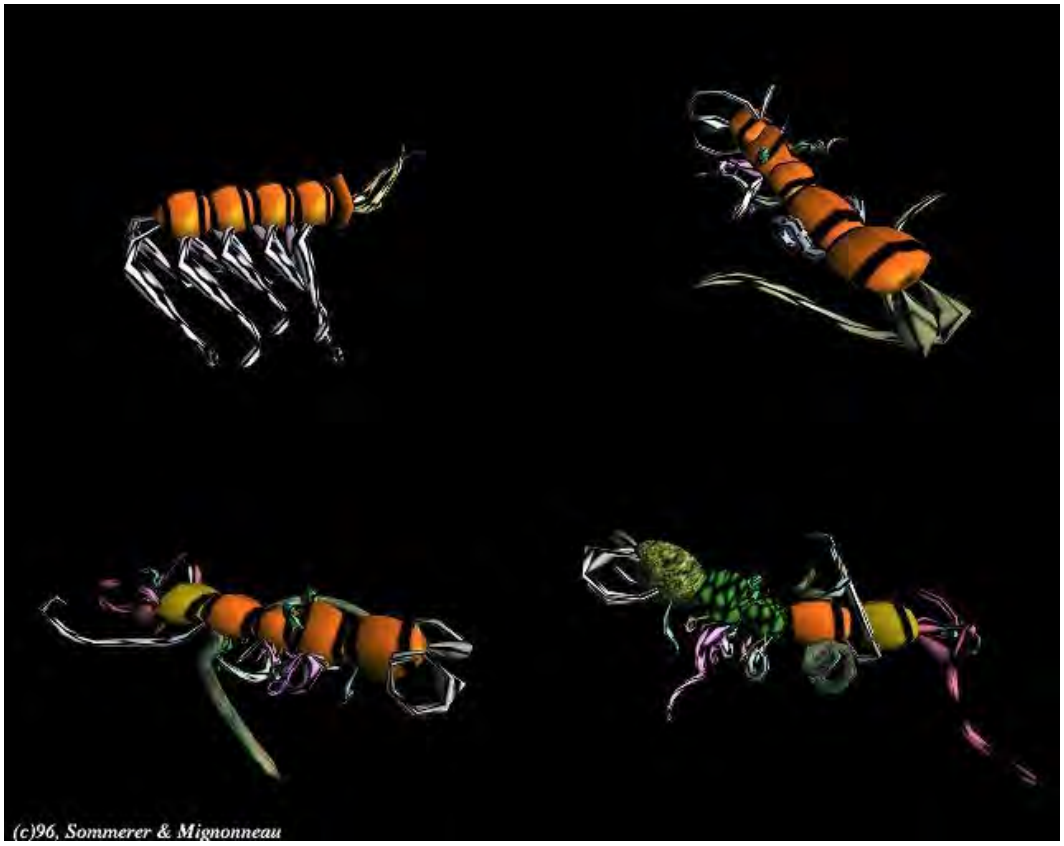


Figure 2.7 - GENMA - Genetic Manipulator by Christa Sommerer & Laurent Mignonneau

"Nature exemplary is represented as artificial nature of a micro scale: abstract amoeboid artificial three-dimensional forms and shapes." [19]

"Principles of artificial life and genetic programming are implemented in those forms or 'creatures', allowing the visitor to manipulate their virtual genes in real time." [19]

We liked the notion that people can manipulate genes and watch the 3D model update in real time. The visuals provide adequate feedback to the user, however, we were not fond of the visual aesthetics.

Chapter III. Methodology

1. Interaction Model

Our thesis allows user(s) to create one of three types of musical creatures at a time. The three creature types are illustrated in Figure 3.2. All three of the creatures have a worm-like body that is made distinctive by color, aural properties and their physical characteristics. The first creature's characteristics are its blue color tones, bass sounds, and rectangular extremities, henceforth referred to as 'Rectangles'. The second creature has yellow color tones, high, short sounds and pointed extremities, 'Spiky'; and the third has red color tones, ambient sounds and flowing tentacles, 'Slinky'.

While in play-back mode, previously made creatures roam about the environment, based on perlin noise [17]. A sample scene is conveyed in Figure 3.1. Creatures interact with one another based on a simple set of rules, which lead to complex behaviors like flocking, fleeing and chasing, explained in further detail later in the paper. The rock, paper, scissors rule set that we chose is assigned in the following manner: Rectangles is greater than Spiky, but less than Slinky; Spiky is greater than Slinky, but less than Rectangles; Slinky is greater than Rectangles, but less than Spiky. This rule set is illustrated in Figure 3.2. This rule set is applied when two creatures are within a variable minimum distance. If the two creatures are equal, they will flock, derived from Craig Reynold's flocking algorithms [18] and Zachary Lieberman's code[9]. When one creature is less than the other, a repulsive force is applied, while the other (greater than) will have a smaller, attractive force to the lesser creature, causing fleeing and chasing behaviors. These forces are based on distance – the further the creatures are from each other, the less the force of attraction.

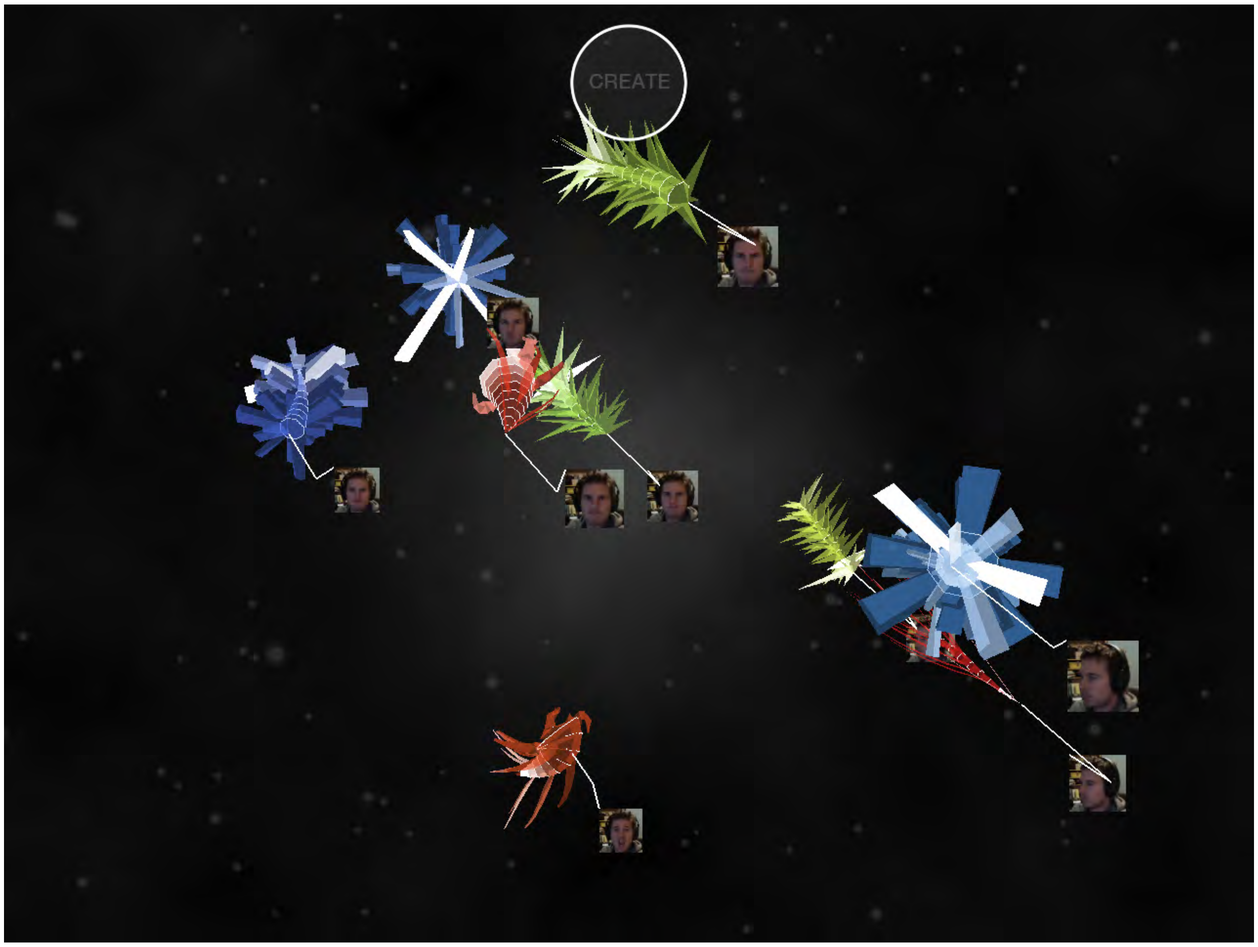


Figure 3.1 - Creatures in play back mode, roaming about the environment. Users may interact with the creatures, moving them about and affecting their aural properties.

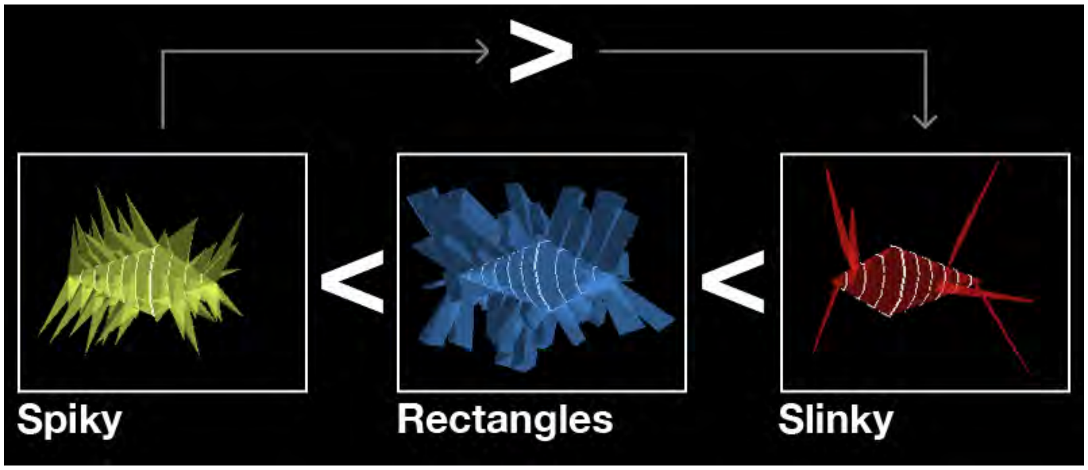


Figure 3.2 – Rock, Paper, Scissors rule set applied to creatures.

Upon approaching the installation, users are provided with a real-time, visual representation of their body movements in the form of a white silhouette in the projection. This allows user's to explore how their body movement affects the scene and their location in the virtual space so that they may adjust their physical orientation to create an ideal interaction. User(s) may disturb the creatures in the virtual space by physically moving their bodies, causing the creatures to repel from their movement (white area), and alter the creature's aural properties to reflect the disturbance. While exploring the virtual and physical environment, a circular button labeled 'create' floats just above the top and center of the detected user(s). This enables people of varying height to reach this button and entices the user to interact with this component. All buttons in this piece are activated through body movement and have an 'energy' or visual feedback that indicates the level of motion detected within the button, the more 'energy', the more opaque the white background is, starting at completely transparent; appearing opaque black. Once the 'energy' has passed a certain threshold, represented by a completely opaque, white background, the button is activated, a sound is generated and its assigned task is performed.

After selecting the 'Create' button, users enter the 'Select Your Creature' scene and are presented with the three creature types in the form of three circular buttons with a sample rotating creature enclosed inside each button. (Figure 3.3). Upon selecting a creature type, the user enters the 'Record Creature Scene' (Figure 3.4), where users are prompted to interact with a pseudo musical instrument composed of eight zones. Each creature is composed of 10 segments, each segment has eight possible musical notes, totaling 80 programmable notes. Users begin by 'programming' the first segment of the creature, the segment's eight notes represented by the eight zones in the instrument, the color of the zones corresponding to the creature type. Users create movement in the zone they wish to activate, which is visually apparent through the 'button energy' mentioned earlier, as well as aurally playing a corresponding note, differentiating for each creature type. Users are given approximately 2.6 seconds to program each of the ten segments of the creature, with the current segment index - represented by a countdown - displayed in the center of screen. A 4/4 (quadruple) simple time signature - also known as common time - is used to maintain a consistent tempo to cycle through the ten segments. This also produces a cymbal sound which also acts as a metronome.

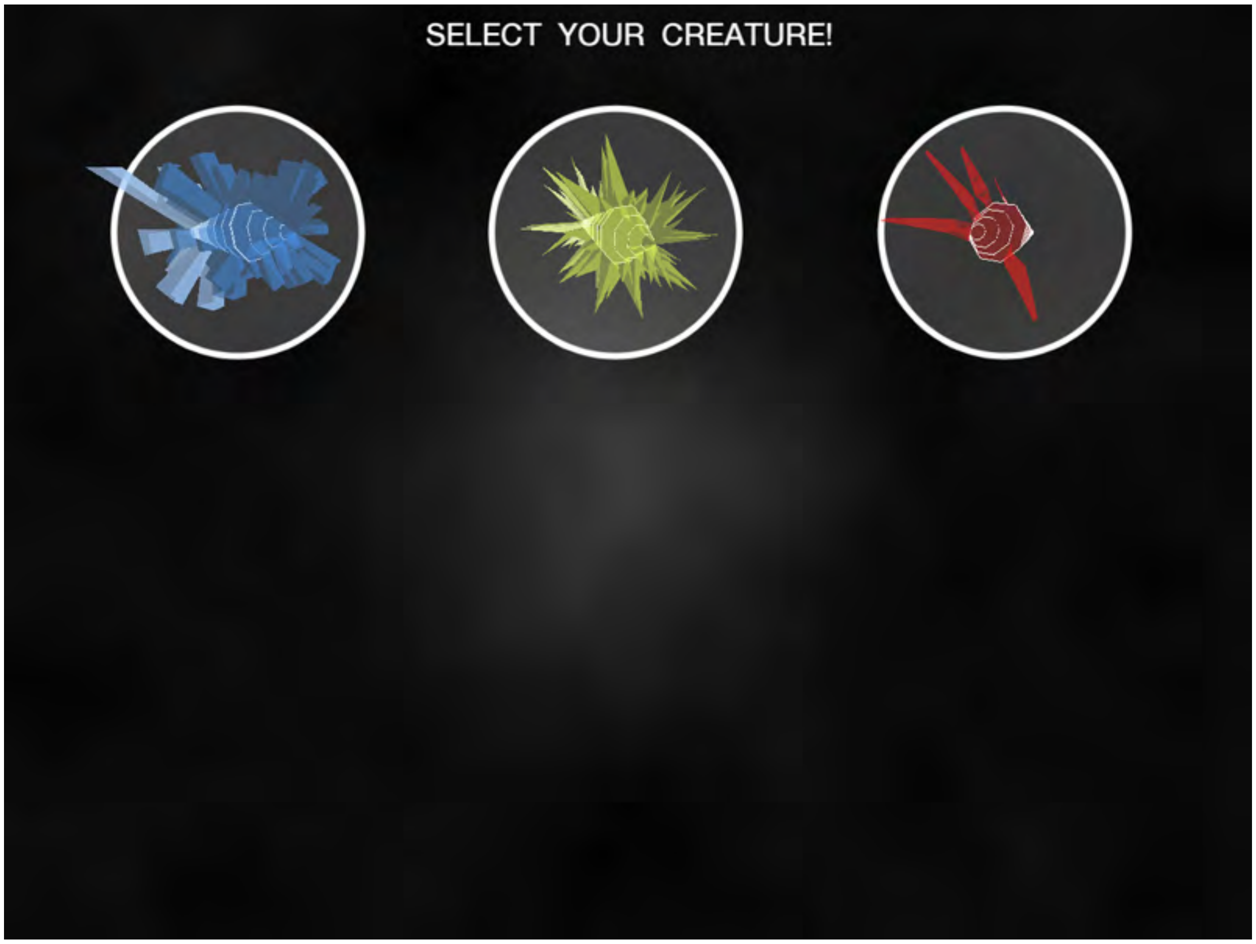


Figure 3.3 - "Select Your Creature" scene. Users may select one of three creature types using body movement.

The user's body movement also determines the speed of playback of the notes and the speed of movement of the creature – the faster the user(s) move, the faster the creature will move. This is reflected visually by the diameter of the creature, the faster the creature, the skinnier it is and vice versa. The user's average movement location also influences the location of the largest ring in the body. If the user moves more on the left of the screen, the largest ring will move to the left, more user movement on the right, the largest ring moves right. These updates occur in real time and are displayed by their creature, displayed right below the count down. (Figure 3.4)

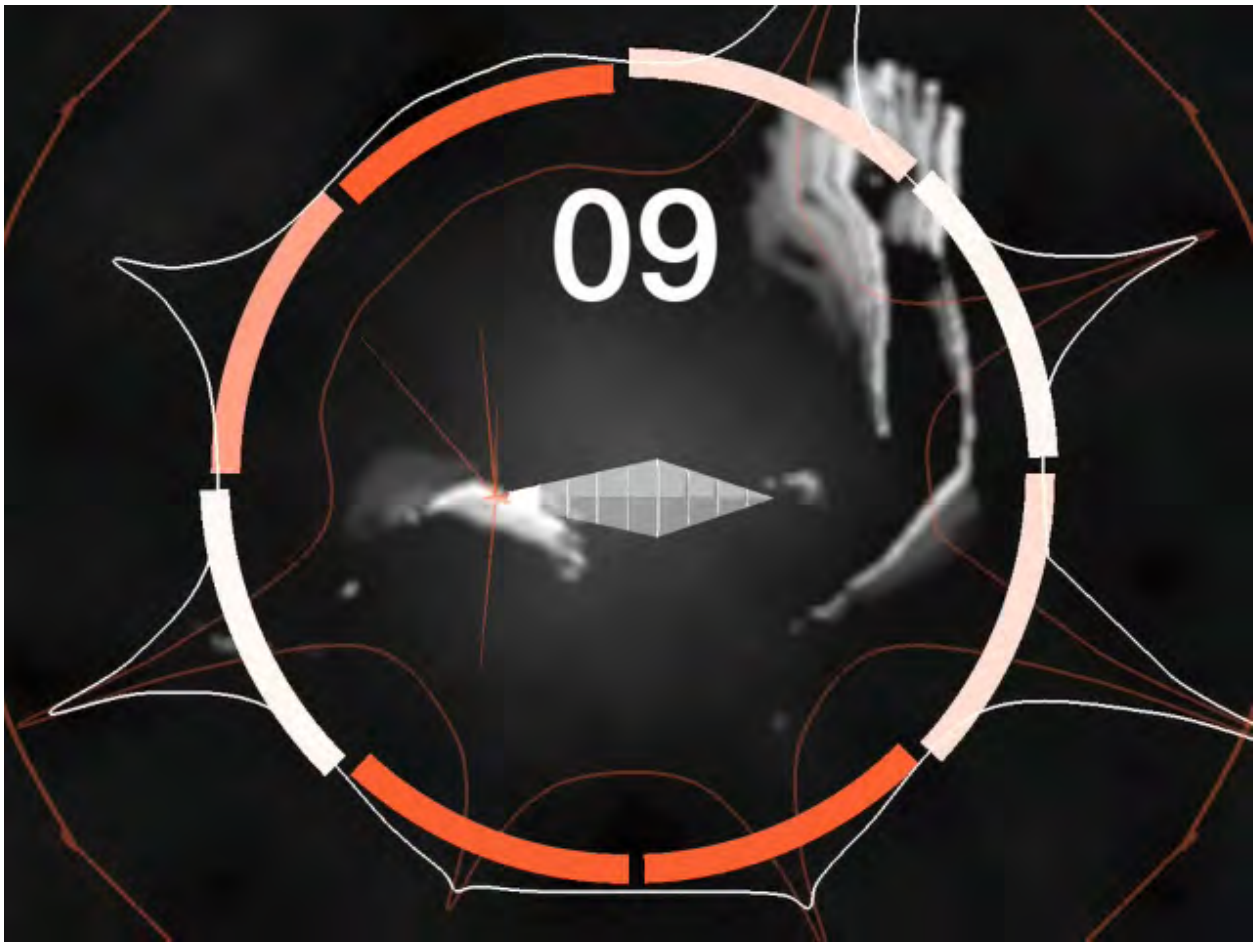


Figure 3.4 - "Record Creature Scene." Users hit zones that activate a note for playback, extend the corresponding extremity and is visually represented by the thin line, the zone turning white and playing its tone.

While recording each segment, the user is also adding extremities to their creature. Every time the user activates a zone on the instrument, its extremity is lengthened, based on the velocity of the motion. The faster the activation motion, the longer the extremity. These extremities are represented by outlines that form based on the type of creature, with the following characteristics: Rectangles is square, Slinky is smooth and Spiky is pointed. Once the segment has been completed, the extremities are attached to the base body (Figure 3.5).

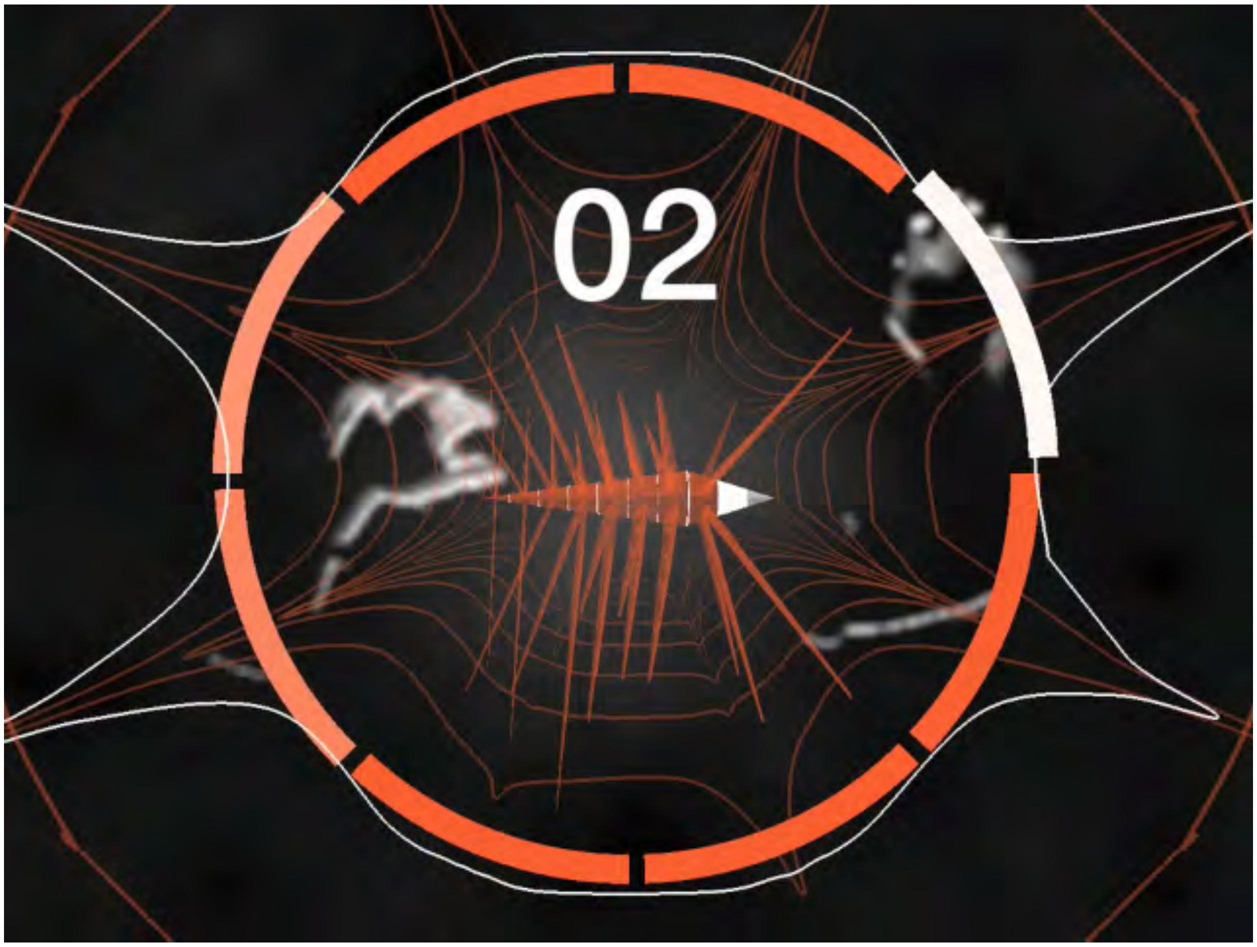


Figure 3.5 - "Record Creature Scene." Creature attributes are updated in real time, based on the user input.

Upon creature completion, users are prompted to 'Smile' and a photo is taken of the user, who may choose to keep the photo or take another, via buttons on either side of the taken photo (Figure 3.6). If the user chooses to take another photo of themselves, video of the user is displayed, instead of the previous image, so that they may position themselves and pose as they wish to be seen by the subsequent viewers. A descending countdown from 3 is displayed just above the video of the user, another photo being taken when the countdown reaches 0. Again, the user may choose to keep this photo or take another. If the user chooses to keep the photo, or does not choose anything for 20 seconds, the current photo is attached to their creature so that it may be easily identified later. The creature is now released into the environment and may interact with previously made creatures.

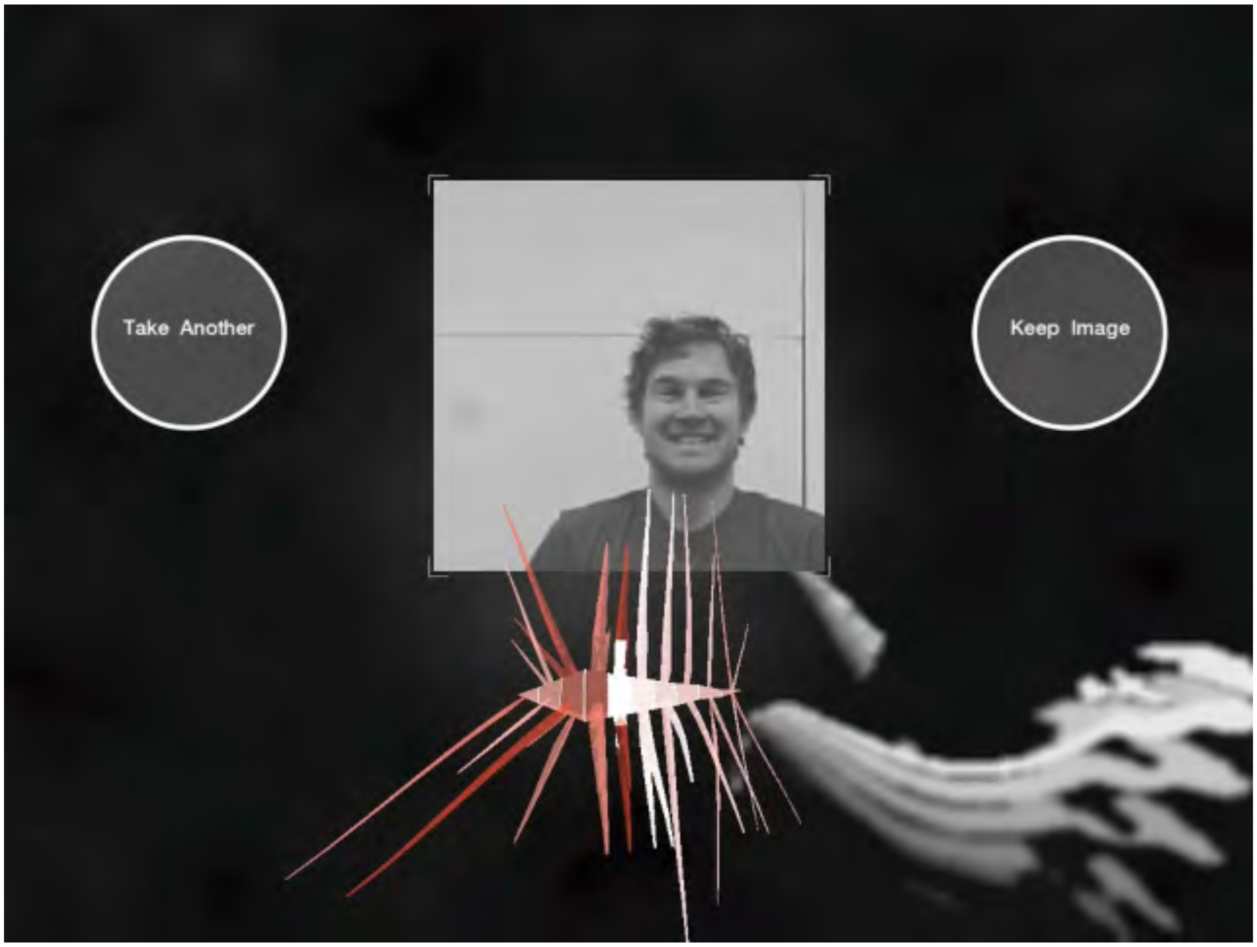


Figure 3.6 - User has just taken a image to be attached to their creature.

More than ten creatures in the environment cause some amount of confusion in the visuals and the audio becomes chaotic. As a remedy, creatures are assigned a life span, the greater the user's motions, the longer the life of the creature, up to a maximum of fifteen minutes. Once the creature's life-span is complete, it slowly dies (Figure 3.7). If the number of creatures in the environment exceeds ten, the oldest creature dies ensuring a balanced environment. The system sustains a minimum of three creatures. If no additional creatures are introduced into the environment, the creatures' lives are extended indefinitely until the introduction of more creatures.

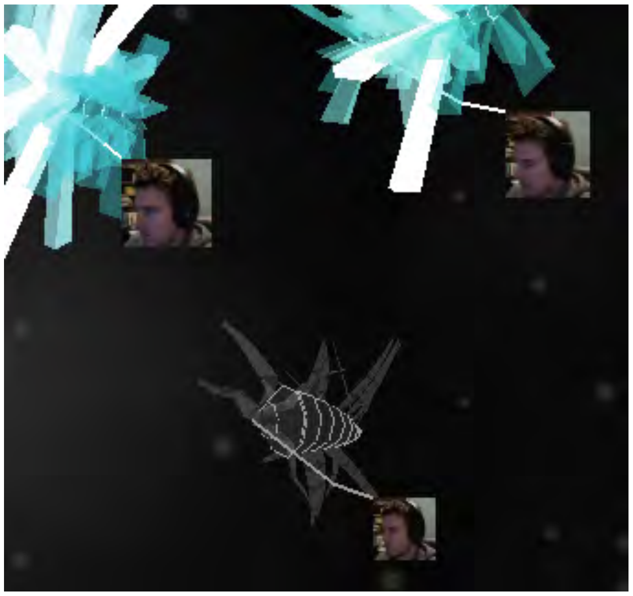


Figure 3.7 - Creature is dying, color fades, its audio volume fades, its movement slows and it will soon disappear from the scene.

2. Installation

We had many considerations for the physical realization of the project, with the final manifestation illustrated in Figure 3.8. The fewer the barriers to interaction, the better. We chose to use speakers, instead of headphones, to allow people to move freely about the space without worrying about wires (if wired), or putting on and replacing the headsets. In addition, users who are acting as the audience are able to experience the audio that the active user is hearing. Utilizing body movement as the main user input, instead of blob tracking openCV (explained later) granted a robust interaction model that worked in variable lighting conditions and resulted in smaller cpu usage. We chose to make a large interaction area, measuring 8 ft. X 8 ft. X 10 ft. to allow for multiple users to engage or disengage as they please. We also wanted the user to make large body movements while creating their creature to ensure an investment in the creature. This was encouraged by using body movement as input, instead of body placement (blob detection).

Furthermore, we wanted to have some control over the visuals and audio, which was solved by limiting the total number of creatures to ten. When a new one is created, it is assigned one of our pre-selected colors for each creature type. This still allows us control over the colors, but gives the creatures a distinctive color, (among other creatures) for easier identification.

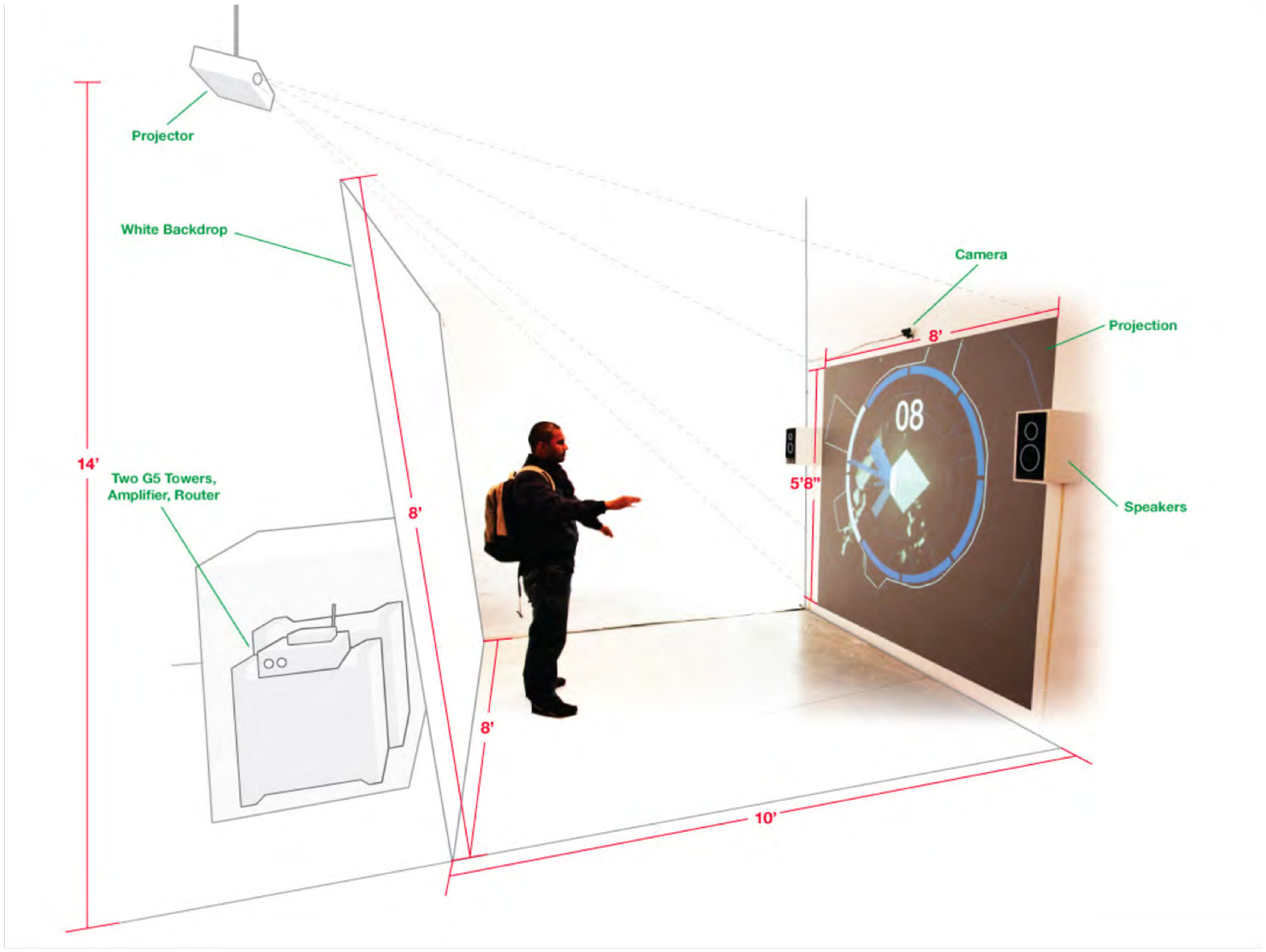


Figure 3.8 - Installation schematics and measurements.

We chose the inexpensive SONY Playstation Eye usb camera which gave us up to 640 x 480 px resolution at 60 fps. The final video capture size was 320 x 240, since we could afford a lesser resolution and it also saved on cpu. We replaced the standard lens with an 8mm lens from a variety pack of standard M12x0.5 thread lenses purchased online. The camera was mounted on the permanent wall, directly above the center of the projection, aiming down at the interaction area. The camera was connected to the computer via a 32.5 ft. usb extension cable, with a small, non-wall powered usb hub connecting the camera cable to extension, to boost the signal. We decided to erect a wall behind the users to block any motion interference that may occur behind the installation. Due to fire restrictions, we were obligated to create a movable wall, which we constructed from two 4 ft by 8 ft sheets of lightweight foam-core, taped together and reinforced with two additional pieces in the back. This 8 ft. by 8 ft. flat surface was held upright by two medium weight lighting stands connected by a crossbar. A lightweight, dark colored cloth was draped over the back of the foam-core to hide the stands, cables and other blemishes. The two computers were placed directly behind this wall and were housed in a wooden podium along with the router, amplifier, audio channel mixer and cables.

3. Implementation

a. Enabling Technologies

The visuals of this project were coded using openFrameworks, "an open source C++ toolkit for creative coding,"[10] and openGL (Open Graphics Library)[14] Some of the code for flocking, particle interaction, and openCV (Open Source Computer Vision)[16] tracking is based on that of Zachary Lieberman [9] The audio is being dynamically generated using SuperCollider [20] There are two Macintosh Intel G5 towers running Mac OSX 10.6 that power the installation. The two computers are placed on a closed wired network with static IP addresses to allow for communication and debugging with laptop computers. One computer hosts the SuperCollider application and the other runs the openFrameworks application. OSC (Open Sound Control) [15] is used to communicate between the two applications. The technology relationships can be seen in Figure 3.9.

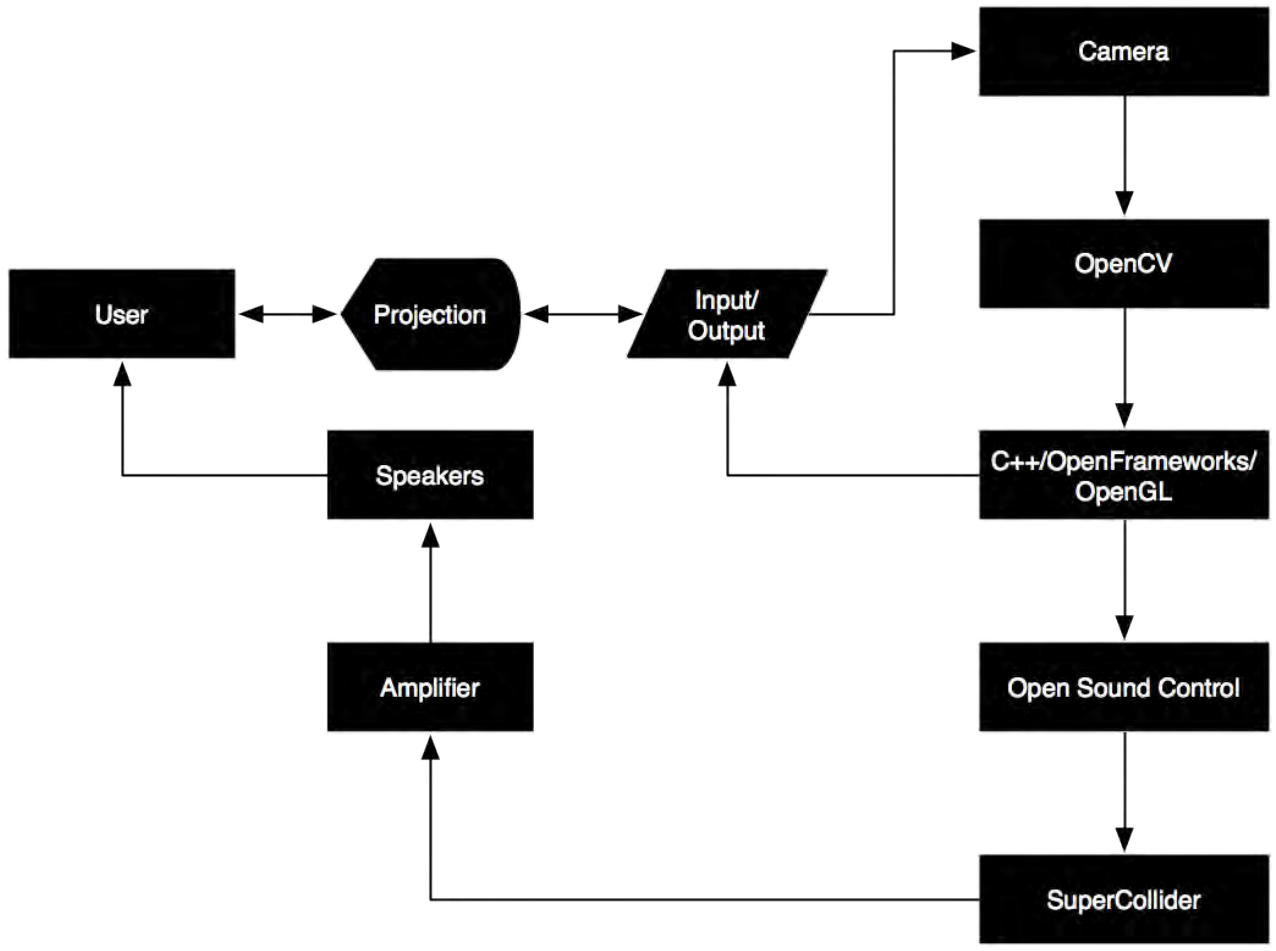


Figure 3.9 - Technology Diagram

b. Visuals

The visuals were programmed in C++, using OpenFrameworks, with an Object Oriented approach. The four different scenes are each given a class that extends a base scene class. The scenes store and control the different objects within them, from the creatures to the user's picture. The next sections will discuss each scene and the organization and set up of the C++ code to achieve the desired visuals.

i. Utilities

The set of classes that are described in this section are used by all of the scenes. These classes include code for the movement tracking and communication with the

SuperCollider program via OSC. A class named ofxBlobTracker is used to track and interpret user motion. This class is based on code by Zachary Lieberman [9]. The motion is detected and stored in a gray image as white pixels, with no motion represented by black pixels. This image serves as a motion history layout and can be accessed by other classes in the program for interpretation. This image is blurred and faded over time to convey motion and is displayed to the user during their interaction as motion feedback.

A class that manages OSC messages was very important in this project. By using a class, we could set the sending port, receiving port and host once and then utilize the class throughout the entire application. This class allowed us to send messages to the SuperCollider application, which interpreted it for sounds for the creature, buttons and other actions discussed in the next section. In addition, this class received messages from the SuperCollider application regarding sound related visuals, such as the advancing of the recording scene visuals.

To determine the amount of motion, the ofxBlobTracker motion image is sent to a Vector Field class, written by Zachary Lieberman. This class interprets the white and black pixel values and converts it into two dimensional forces.

ii. Render Creature Scene

The render creature scene controls all of the user - created creatures, the camera and the ambient particles. The base creature class contains functions for drawing, updating and flocking. The update function moves all of the base body point locations based on outside forces that have been applied. The flocking functions apply the forces that have been added based on the location of its neighbors. The draw function renders the body of the worm. Each type of creature, Rectangle, Spiky and Slinky, has an accommodating class that extends the base creature class, and contains update and draw functions that add functionality to the base class. The update functions pertain to the extremities of the creature and apply the physics accordingly. The draw functions contain code that renders the extremities using OpenGL.

iii. Select Creature Scene

This scene is simple, with one creature of each type that generates unique, random extremities on each creature every time the scene is accessed. It sets each creature on a rotating, three dimensional axis with a circle in the background that acts as the button.

iv. Record Creature Scene

This class contains an array of objects that contain points that the user will manipulate and ultimately use to determine the properties of the extremities. The main class receives OSC messages from the SuperCollider application, which calls a function within the record scene class that advances the segments, which contain the recording points. The points are manipulated using the Vector Field class discussed earlier.

v. Claim Creature Scene

The claim creature scene uses the video images captured by the ofxBlobTracker class. Each creature has a texture assigned to it, and when the user keeps an image in this scene, the texture is attached to their creature. The creature's movement into the scene is accomplished by moving the head of the creature, so that the creature is positioned accordingly and conveniently for the viewer.

c. Sound

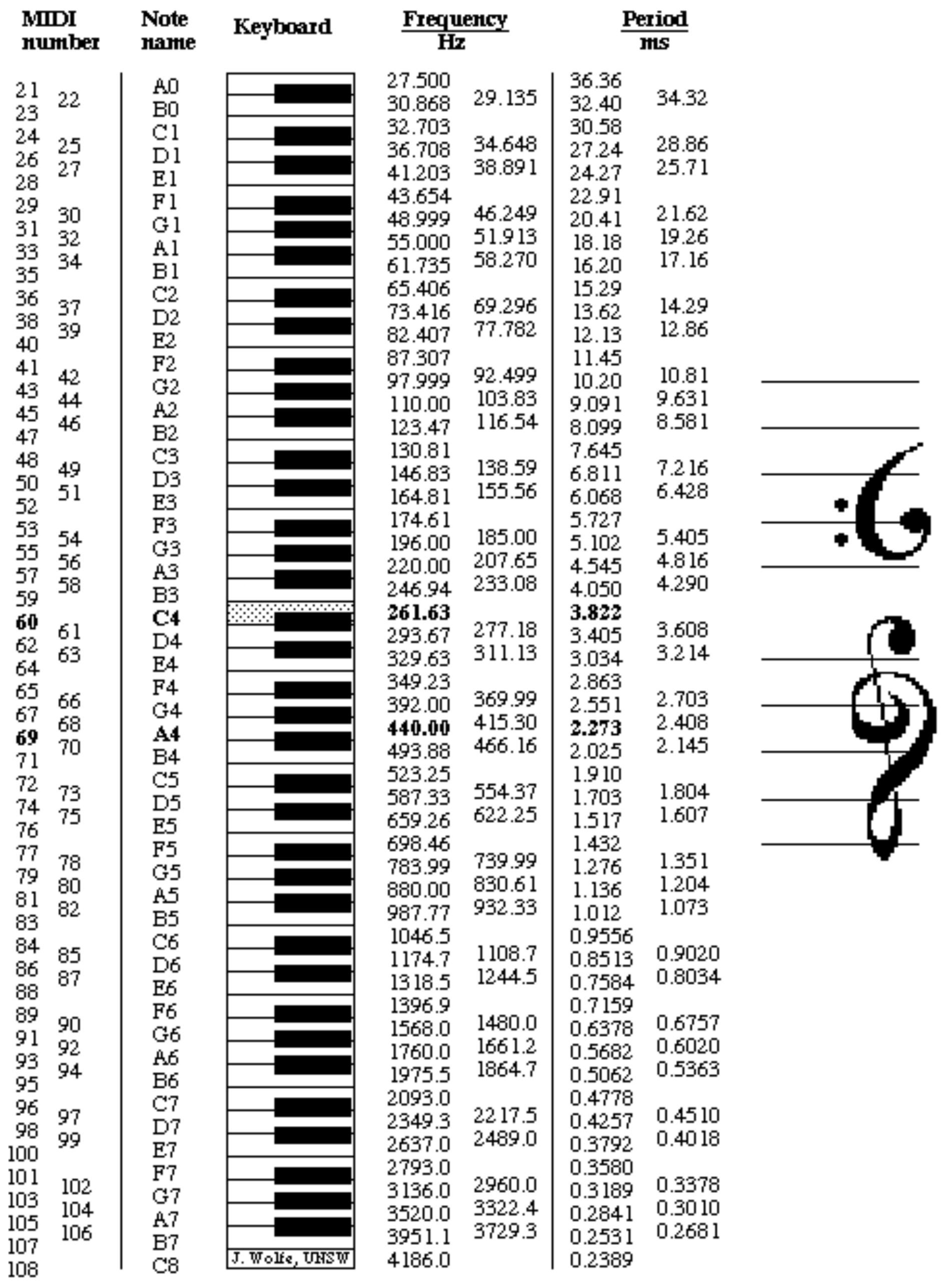
The sound was programmed in SuperCollider [20] based on an Object Oriented Programming model [reference]. The most important part of sound was to create characteristic tones for each creature. This proved to be a challenging problem since we also wanted the sound to maintain a harmonious sonic structure while being able to dynamically control certain parameters and maintain the individualistic nature for each creature. Besides sounds for the creatures, the program also includes sounds for the interface buttons and a metronome. Described below are the important components regarding the approach and implementation to achieve the desired results.

i. Scales and notes

Each creature has it's own eight-note scale which differentiates the way it sounds from other creatures including creatures of the same type. Each time a creature is created, a random scale is picked from a predetermined selection - minor pentatonic, major pentatonic, ionian, dorian, phrygian, lydian, mixolydian, aeolian, locrian, harmonic minor, harmonic major, melodic minor and diminished.

After a scale is selected, eight notes are generated based on a base frequency. The base frequency is determined by randomly generating a number in a specific range, which is used as a MIDI number, figure 3.10 shows the relationship between note names, MIDI numbers and frequencies:

- Rectangles: between 33 - 45 (55.0Hz - 110.0Hz), to yield low frequency/bass notes
- Spiky: between 57 - 69 (220.0Hz - 440.0Hz), to yield notes of higher frequencies
- Slinky: between 45 - 69 (110.0Hz - 440.0Hz), a wider range from low to high frequencies



MIDI number		Note name	Keyboard	Frequency Hz		Period ms	
21	22	A0		27.500	29.135	36.36	34.32
23		B0		30.868		32.40	
24	25	C1		32.703	34.648	30.58	28.86
26	27	D1		36.708	38.891	27.24	25.71
28		E1		41.203		24.27	
29	30	F1		43.654	46.249	22.91	21.62
31	32	G1		48.999	51.913	20.41	19.26
33	34	A1		55.000	58.270	18.18	17.16
35		B1		61.735		16.20	
36	37	C2		65.406	69.296	15.29	14.29
38	39	D2		73.416	77.782	13.62	12.86
40		E2		82.407		12.13	
41	42	F2		87.307	92.499	11.45	10.81
43	44	G2		97.999	103.83	10.20	9.631
45	46	A2		110.00	116.54	9.091	8.581
47		B2		123.47		8.099	
48	49	C3		130.81	138.59	7.645	7.216
50	51	D3		146.83	155.56	6.811	6.428
52		E3		164.81		6.068	
53	54	F3		174.61	185.00	5.727	5.405
55	56	G3		196.00	207.65	5.102	4.816
57	58	A3		220.00	233.08	4.545	4.290
59		B3		246.94		4.050	
60	61	**C4**		**261.63**	277.18	**3.822**	3.608
62	63	D4		293.67	311.13	3.405	3.214
64		E4		329.63		3.034	
65	66	F4		349.23	369.99	2.863	2.703
67	68	G4		392.00	415.30	2.551	2.408
69	70	**A4**		**440.00**	466.16	**2.273**	2.145
71		B4		493.88		2.025	
72	73	C5		523.25	554.37	1.910	1.804
74	75	D5		587.33	622.25	1.703	1.607
76		E5		659.26		1.517	
77	78	F5		698.46	739.99	1.432	1.351
79	80	G5		783.99	830.61	1.276	1.204
81	82	A5		880.00	932.33	1.136	1.073
83		B5		987.77		1.012	
84	85	C6		1046.5	1108.7	0.9556	0.9020
86	87	D6		1174.7	1244.5	0.8513	0.8034
88		E6		1318.5		0.7584	
89	90	F6		1396.9	1480.0	0.7159	0.6757
91	92	G6		1568.0	1661.2	0.6378	0.6020
93	94	A6		1760.0	1864.7	0.5682	0.5363
95		B6		1975.5		0.5062	
96	97	C7		2093.0	2217.5	0.4778	0.4510
98	99	D7		2349.3	2489.0	0.4257	0.4018
100		E7		2637.0		0.3792	
101	102	F7		2793.0	2960.0	0.3580	0.3378
103	104	G7		3136.0	3322.4	0.3189	0.3010
105	106	A7		3520.0	3729.3	0.2841	0.2681
107		B7		3951.1		0.2531	
108		C8	J. Wolfe, UNSW	4186.0		0.2389	

Figure 3.10 - MIDI numbers and their corresponding frequencies. [Source: http://www.phys.unsw.edu.au/jw/notes.html]

ii. Program Structure

This section describes the two largest most important functions of the program.

Foundation

The first time the SuperCollider program is executed, it lays the foundation that enables dynamic control over the process of creating and manipulating all the components within the program. The flow of the program is shown in figure 3.11 with a brief description of each step below, the source code can be found under Appendix B:

- Create list to hold creature objects: A list is an empty container that holds all dynamically created creature objects.
- Create a Scale Generator object: The scale generator object contains predefined scales based on which notes are generated.
- Store SynthDefs to virtual memory: All tones in the program are considered to be an instrument by itself which are called SynthDefs within SuperCollider. Loading these abstractions into virtual memory enables them to be reused over and over without overloading memory.
- Create a Limiter: A limiter is applied to all the signals that are output. It allows signals below a specified input power to pass unaffected while attenuating the peaks of stronger signals that exceed this input power [11].
- Create net address to receive OSC messages: This net address is an IP address of the computer to which Open Sound Control messages are to be sent.
- Create OSC responder object: The OSC responders object provides methods via which the program can be controlled.
- Create net address to send OSC messages: This net address is an IP address which receives OSC messages. These messages are used to control various aspects of the SuperCollider program.
- Create a list to hold all the buttons: A container that holds instances of all the buttons that are part of the interaction.

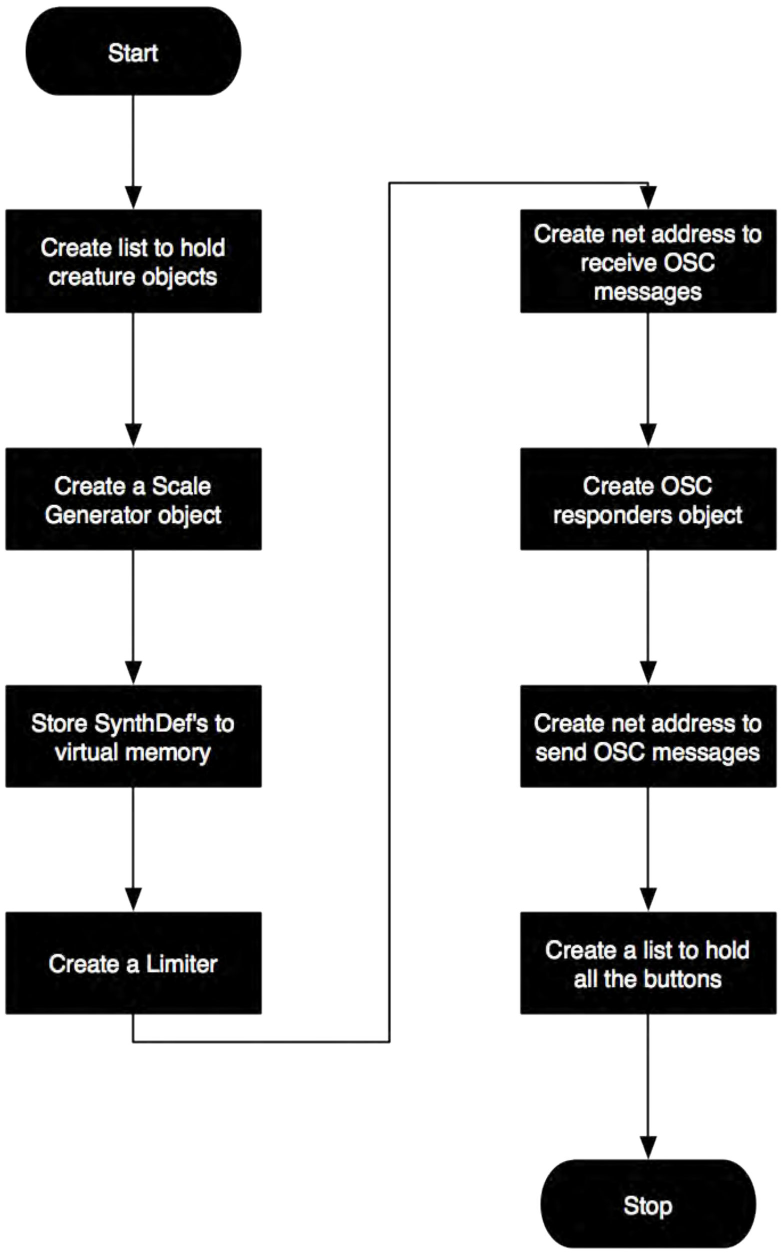

Start
Create list to hold creature objects
Create a Scale Generator object
Store SynthDef's to virtual memory
Create a Limiter
Create net address to receive OSC messages
Create OSC responders object
Create net address to send OSC messages
Create a list to hold all the buttons
Stop

Figure 3.11 - Foundation

Create creature

Creating a creature is the primary function of the program. All other features and functionality is tied into this function and are brought into effect once a creature is created. The flow of this is shown in figure 3.12 with a brief description below:

- Select creature: The participant selects the desired creature from the interface described in the interaction model earlier in this chapter.
- Set creature tempo to world tempo: This step helps maintain the tempo of the new creature in sync with other creatures in the environment.
- Generate and assign an eight-note scale: An eight-note scale is generated and a note is assigned to each of the triggers in all ten segments. These are stored in an array which serves as a look up table for playing back single or multiple notes.
- Play and store triggered note: The assigned notes are called and played back by the C++ program. Notes triggered by the participant are stored by the C++ program as an on or off event (0 or 1).
- Build sequence with stored notes: The triggered notes are stored as an array of 0's and 1's which are then sent back to the SuperCollider program via OSC. The SuperCollider program stores this array and prepares to play back the notes from the lookup table as a compound cluster or chord for each creature segment.
- Assign tempo to play sequence and playback: The C++ program assigns the tempo to the creature, based on the participants movement, and triggers the playback.

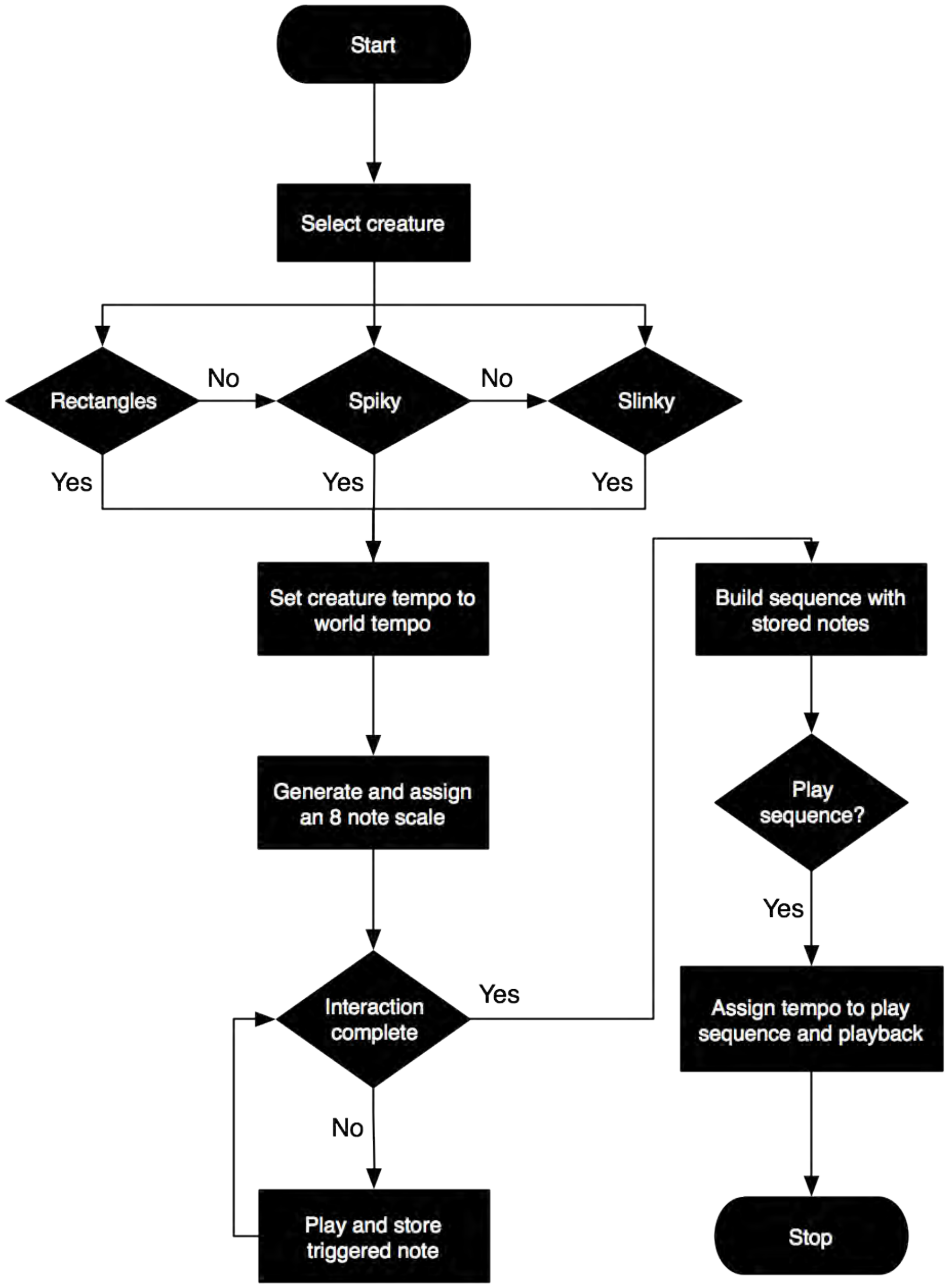

Start
Select creature
Rectangles
No
Spiky
No
Slinky
Yes
Yes
Yes
Set creature tempo to world tempo
Generate and assign an 8 note scale
Interaction complete
Yes
No
Play and store triggered note
Build sequence with stored notes
Play sequence?
Yes
Assign tempo to play sequence and playback
Stop

Figure 3.12 - Create creature

iii. **Classes**

This section describes the important classes in the program.

Responders

This class contains methods that are made available to be controlled via Open Sound Control and is the most important class in the program. This class contains a series of OSCResponders with specific addresses which enables us to create a custom protocol. The custom protocol was created to easily trigger specific functions within the SuperCollider program from the C++ program via OSC messages. Each method has a specific protocol and parameters that need to be passed to it. For example to create a new creature the message would look like [/createCreature, type, id]. The properties and methods of the class are shown in figure 3.13. The method names are self explanatory as to what they do.

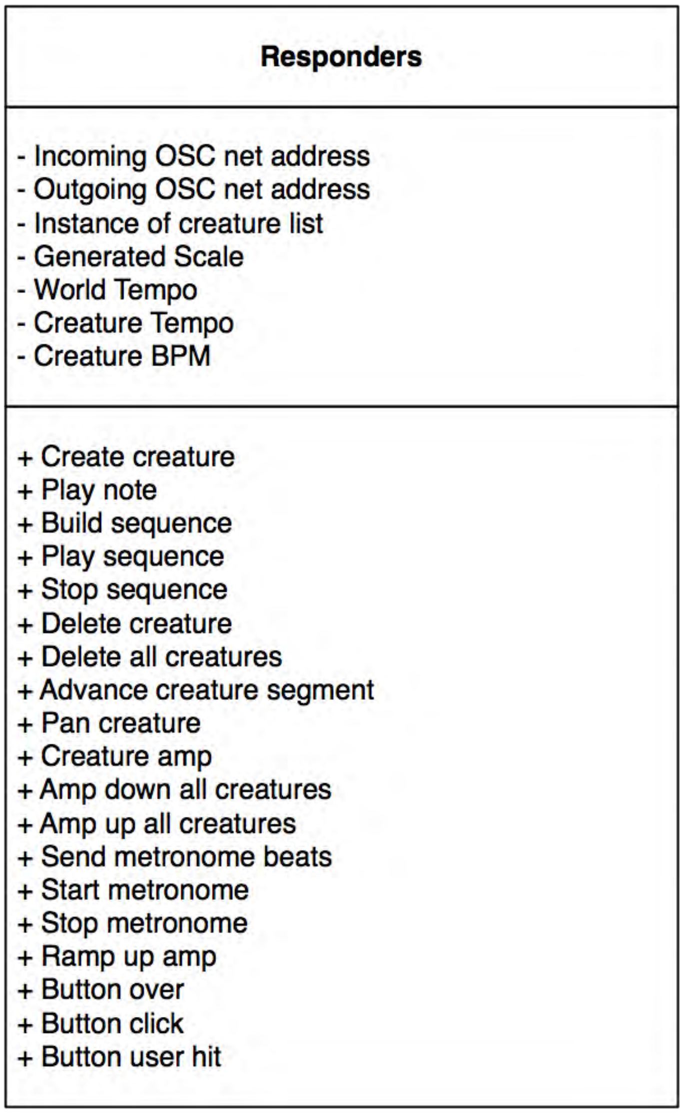


Figure 3.13 - SuperCollider Responders Class

Creatures

Each of the three creatures was considered to be an instrument called a SynthDef - in SuperCollider terminology. The SynthDef defines the characteristic sound for the creature and includes parameters that can be manipulated dynamically. Each creature was created as an object with different properties and methods that enables individual control over each creature. The properties and methods for the three creatures is shown in figure 3.14.

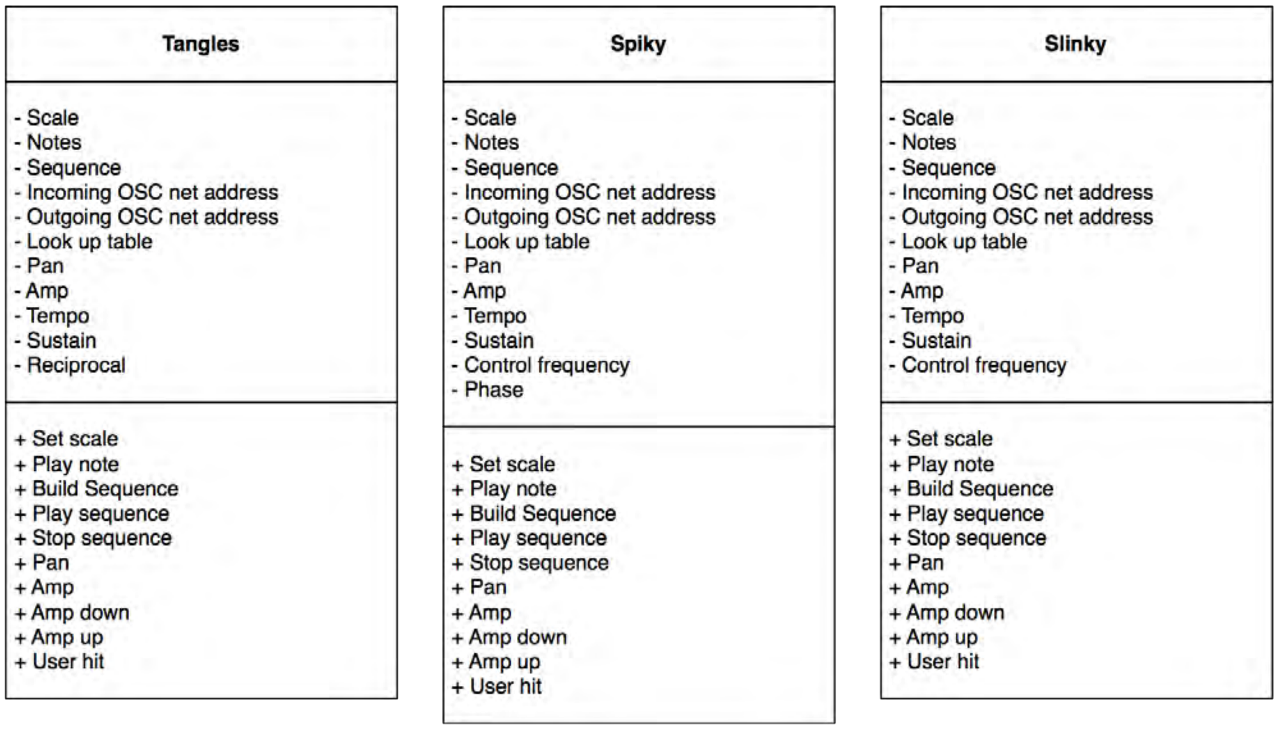


Figure 3.14 - SuperCollider Creatures classes - Tangles, Spiky and Slinky

The pan, amp up, amp down and user hit are methods that are controlled dynamically. The pan is based on the position of the creature on the screen and pans the audio across the left and right channel accordingly. The amp up and amp down methods are used to reduce or increase the volume of the creature. This is done on two occasions:

- When participants are creating a new creature.
- To slowly ramp up the volume after the creature is created.

The user hit effects specific parameters for each creature type to alter the sound:

- The rectangles creature sounds like a bass instrument. When a participant interacts with the creature, the reciprocal is altered based on the values from the C++ program via OSC, thus changing the 'ring' or 'clicking' sound.
- The spiky creature sounds like a piano and contains a Decimator effect. The control frequency and phase parameters of the effect are altered when a participant interacts with this creature. This distorts the sound by reduction of the resolution of the audio data. This digital audio effect is also known as a Bitcrusher.

- The slinky creature sounds like a warm synthesizer. When a participant interacts with this creature is changes the control frequency based on the values received thus altering the timbre of the sound.

4. User Scenarios

Described below are user scenarios to further demonstrate the functionality and flow of the installation.

Scenario 1

Matt is a young new media artist who earns a living by making creative applications in Flash with AS3 for deployment on the web. He has experimented in his free time with ideas of motion tracking and understands how to structure code for building medium-sized applications. When he approaches the installation, he first observes the movement of the creatures and briefly contemplates how the creatures are being rendered. He immediately begins to explore the physical space and quickly discovers his motion feedback and its affect on the creatures in the environment. He proceeds to make a creature and chooses the Slinky type because of its free flowing tentacles. He quickly grasps the functionality and his abilities while creating his creature. He initially waves his hands and moves his body erratically, but decides to try to make a musical rhythm by equally timing the hitting of different notes. When his creature is complete he makes a semi-serious face and immediately decides to keep his creature. He watches as the creature enters the scene and he chuckles as his picture is being dragged behind. He continues to interact with the creatures, revisiting his prior thoughts about the rendering of the creatures. He hypothesizes how the creatures move about the environment, how the camera follows them and how the audio is being generated.

Scenario 2

Jean owns a computer and casually uses it to check emails and view family photos online. She is interested in all types of art and is drawn to the installation out of curiosity. She stands off to the side and watches the creatures move about the virtual environment, but is unaware that it is interactive. A young mother and child walk through the interaction area and notice their silhouettes and disturbance on several creatures, which peaks the interest of the mother, child and onlooker Jean. The mother and child move about, exploring the relation between their physical movements and its result in the virtual space. The child approaches the projection surface and attempts to touch the creatures displayed on the wall. The mother instructs her child to come closer her as she stands near the temporary wall. She demonstrates hitting multiple creatures, and they continue to move around and finally discover their ideal location in the physical space near the temporary wall. The mother asks the child "what's this?" and then selects the 'Create' button by making large sweeping arm motions. They proceed to create a Spiky character, the mother moves her legs while holding the child who waves frantically, and Jean gladly observes. The mother constantly instructs the child to 'hit' the different zones and does so herself. Their creature is complete and they smile, as instructed by the installation, the child laughs, but is ultimately unsatisfied with the image. The mother activates the 'Take Another' image button and the countdown for another photo begins. The mother tells the child to smile and they embrace one another in pose, Jean quietly onlooking and admiring. They choose to keep this photo and the mother instructs her child to look at the creature they just created. The child tries to influence the direction of their creature by waving his arms and tells his mom to look at the photo of them. Jean is amused by the child's enthusiasm and is satisfied watching others interact with the project, discovering the functionality through others experimentation.

5. User Testing

We performed a user test on March 11, 2010 on two users in the same field of study, challenging our concept and implementation. Users were presented with our thesis concept at the time:

"[Thesis] is an art piece that abstractly references data ownership, privacy and its future in the form of an interactive audio / visual installation. The user's movements are recorded by a camera and used to affect the physical, kinematic and auditory properties of a single, worm-like organism. The organism moves about in a virtual environment and evolves over time as more users interact with it."

More information, including the testing protocol, feedback, images and more in-depth analysis can be seen in the attached document, Appendix A.

The following is a summary of our analysis:

This user testing was extremely useful and provided valuable feedback that we used to mold and refine our current concept. The idea that users are prompted to perform an activity was interesting. Even something as simple as a countdown of how much time is allowed for the interaction will convey much needed feedback to the user.

We are very fond of the idea that people can create their own creatures and watch them throughout the environment and will most likely implement this for our project. This is very close to the social networks that we were referencing. We are discussing whether the user should be able to choose which color (personality) they want to create. As long as people know which creature is theirs, this should retain interest in the piece beyond the initial interaction as the creatures will interact with each other.

The recommendation for using the Rock, Paper, Scissors method is a great example of simple logic that can lead to complex behaviors of the sum of parts. We will research this further, and will determine if this will fit into our creatures.

While implementing the project, we constantly asked colleagues for feedback relating to the interaction model. Initially we had a static camera, the creatures moved about a fixed space, wandering in and out of visibility, with a slight force towards the center to ensure they do not roam too far. We programmed shortcut keys to allow for quick creation of the three types of creatures. We created several creatures and showed it to a potential user, who suggested we make the environment more dynamic by having the camera follow the creatures. We decided to implement this suggestion, but have the camera only dolly from side to side, and up and down, maintaining a constant orientation.

We often set up the installation in a common space so that colleagues who were willing could test our interaction model and flow. Based on one users feedback we implemented a metronome so that people could keep the beat. We presented the project near completion to a large group of users simultaneously for feedback pertaining to the recording of the creature and its comprehensiveness. Some of the users felt it was unclear what they were supposed to interact with, referring to the pseudo musical instrument. For this reason, we animated the triggers in from gray to color and simulated each being hit in a delayed sequence, while displaying the text "Hit the triggers to build your creature." They also suggested that the user's manipulation of their creature be more apparent. Based on this feedback, we added the creature currently being modified in the center of the recording scene, updated in real time. After explaining that on each beat, a new segment in the creature is being manipulated, the users felt confused and it did not feel intuitive. This is why we decided to add a transition effect to make enhance the notion of the user traveling through the center of the creature and adding extremities.

While setting up, testing and calibrating for the gallery space, we encountered numerous users creating creatures and interacting with them in the environment, allowing us to fine tune the settings and accommodate for a wide range of users. A final suggestion, just days before the show, was to simply add the text smile while taking the initial photo, as a call to action. This proved very successful as most users did in fact smile and most were prepared for a photo to be taken.

Chapter IV. Evaluation

1. Results

The project was successful at a variety of levels as described below.

a. Idea and Form

The idea was for participants to interact with our project to create their public identity. The awareness in creating this identity would be dictated by the amount of physical motion they were willing to invest. A majority of the participants gesticulated wildly to create their creatures while others did the bare minimum. All participants engaged in taking their pictures and proceeded to observe how their creatures interacted with other creatures in the environment. They expressed awe and wonderment at their creation and many returned to observe the state of their creature after a while. Many participants created multiple creatures exploring how and to what extent their physical movement would effect the physical qualities of the creatures. Some participants exhibited performative behavior by dancing and kicking in a synchronous manner while others chose to be an audience watching and encouraging the 'performers'. Many participants enjoyed interacting with the creatures in the environment pushing and flicking them around.

b. Types of users

We did get some insight into the different types of participants and their behaviors, although this was not a central part of our study:

i. Performers and explorers

People who danced, kicked and liked being the center of attention. People who explored how their body movements affected the creature.

ii. Narcissist
People who really liked to take good pictures of themselves.

iii. Audience
People who stood by and enjoyed watching others interact with the project - some of these people were also shy to actually take center stage and interact with the project. They also encouraged participants.

iv. Trainers
People who encouraged people to participate. They also demonstrated or explained what the project was and how to interact with it.

c. Interaction/Experience

One of the core challenges while developing this project was how to make an interface that would be inviting and easy to use. Besides building and intuitive interface it huge task was to get it to work well in the physical space. After numerous user tests it was evident that people were trying to grab and hit the creatures. We did eventually build in this feature after some tests and observed that a majority of the participants showed more interest in building their own creature. Most participants knew what to do almost immediately while others learnt by watching and some were helped by other participants. They naturally tried to interact with the creatures by waving their arms and moving about before proceeding to create their own creature. Some participants took a few seconds to orient themselves in the space.

We provided many cues to help orient participants and enable the interaction which we arrived at from user testing, best practices and studying various precedents. These were successful and provided for a satisfying user experience:

- Users could see their silhouette in the projection which gave them a sense of them being a part of the environment and also helped orient them in the space.

- The create button which was displayed as a bubble floating at the top of the projection and moved about based on the location of the user.
- A sentence that indicated how to use the creature creation interface and a short animation that highlighted the triggers.

We accounted for multiple people interacting with the project which occurred at certain times. Small groups of people collaborated to create creatures and took group pictures to attach to their creature. Although this was not as common as a single person interacting with the project. On the opening night people queued up to interact with the project mostly one person at a time. One person would interact or create their creature and step aside for the next person. It was reminiscent of a stage performance or talent show. This was an unexpected behavior which was very interesting to observe.

d. Significance

For participants it proved to be an engaging and enjoyable experience which is the primary goal of our project. The underlying layer of the experience bears a relationship to studies of how users like to create and customize their avatars in virtual spaces. Our project takes this experience and adds a 'physical' dimension to it. Users have to put physical effort to create their 'avatars'. The physicality of the interaction makes them more invested in their creations and resulting in a satisfying experience and larger emotional attachment to their creation. The experience also helped bring out performative and collaborative behavior in the participants which exemplifies how a physical space can be used to effect human behavior.

e. Contribution

i. Algorithmic art

The project is built with a melange of open source technologies to produce a highly stylized, interactive, three-dimensional, audio-visual, environment. The project questions the limits and pushes the boundaries of how these technologies are currently used together to produce algorithmic art and installations.

ii. Human computer interaction

The intention of the project was to create an engaging audio-visual experience that based on our domains of interest. The key for the success of the project was to engage the audience to participate in and be a part of the experience. Many participants view the project as a game while others view it as an artistic expression. We would like to view it as an experience that explores the boundaries between the engaging nature of games, the visual expressiveness algorithmic art and the social nature of human beings to interact with the piece. The project make use of living systems algorithms, audio signal processing and synthesis to produce a dynamic audio and visual experience that aims to make the participant a central figure in the experience.

iii. Data

The project approaches data in an alternative way. Users creating their public identity is most commonly perceived as their online public profiles. This project abstracts this phenomenon and relates it to a physical public space. The project takes advantage of the fact that public profiles can be superficial and possibly fabricated. The project takes this behavioral data thus establishing a relationship between a participant and the space through a virtual 'window'. It captures the emotion of the users in that time and space. The data augments the space giving it a new meaning while the user leaves with a sense of attachment to the space.

f. Design and Technology Components

The project involved a variety of design and technology components to achieve and/or solve specific challenges. The main components are briefly discussed below. A more detailed explanation and usage is described under the methodology section.

i. C++/OpenFrameworks [10]

The interaction, animation, assimilation and dissemination of data was programmed in C++ using the OpenFrameworks library [reference]. An Object Oriented Programming (OOP) [reference] approach was used to create a modularly structured, scalable and flexible program. We chose to use C++/OpenFrameworks over other tools - like Processing or Max/MSP/Jitter - because it is a high performance and robust framework which offers a low level of control. With the variety of processor intensive tasks that needed to be performed this seemed like the appropriate option. It also lent itself to efficient memory management and direct access to computer hardware - video card and sound card.

ii. Open Source Computer Vision (OpenCV) [16]

The OpenCV library was used for motion tracking against a white background. It was also used to for blob detection. This was a key component to capture motion data to make for a successful interaction. This component was the hardest to work with since it has a direct relationship to the space and lighting conditions in the space. To achieve this we spent a lot of time working in the space tweaking and adjusting the responsiveness of the motion capture.

iii. Open Graphics Library (OpenGL) [14]

OpenGL was used to render the three-dimensional environment which includes drawing and rendering the creatures, their animation in the environment and the motion of the virtual camera. Numerous tests and optimizations were made to achieve effective performance and rendering. Certain limitations were also built into the program to achieve a desired and consistent frame rate.

iv. SuperCollider (SC) [20]

SuperCollider was used to generate dynamic audio for creatures and effects. The SuperCollider program deciphered the data received from the C++/OpenFrameworks program and generated notes, scales and tones based on the type of creature. Parameters to control the tempo, timbre, pan and amplitude of the sound were also built into the program. An OOP approach was used to create this component as well. We chose SuperCollider over other tools - like Max/MSP and Reaktor - for the same reasons we chose C++/OpenFrameworks - performance, low level of control and efficient memory management.

v. Open Sound Control (OSC) [15]

OSC was used to send data in the form of messages or packets from the C++/OpenFrameworks program to SuperCollider and vice versa. This tool was used because it afforded high performance and low latency data exchange across a network. This enabled the video and sound to be a seamlessly integrated experience with a very low level of latency.

2. PROCESS

From a technical standpoint, we were already familiar with all the technologies used to produce this project. We, however, wanted to dig deeper into C++/OpenFrameworks and SuperCollider and how to use them to create a large audio-visual experience. Throughout the process new techniques were learnt and implemented. Working on this project really armed us with advanced knowledge of the tools we chose and how to apply them to the project. Another key area of learning was usage of a projection that we had no previous experience working with.

a. Iterative process

Throughout the project we followed and iterative design process along the lines of the model described in "What do prototypes prototype?"[6]. The model is broken down into three dimensions - role, look and feel and implementation. Each of the dimensions corresponds to specific questions that we needed answered during the iterative process. We brainstormed the challenges and design questioned we needed to answer before building a prototype and user testing it to gain more insight and feedback from users. This section describes the key brainstorms and prototypes that helped empower our ideation process.

i. Brainstorming

Brainstorming proved a very effective way to generate ideas and come up with solutions to specific problems. It also helped us identify problem areas and outline the workings of our entire system. Brainstorms were conducted using low fidelity methods like sketching on a whiteboard, using post its and also using online brainstorming tools like twiddla for brainstorming remotely [22]. Figures 4.1 - 4.3.

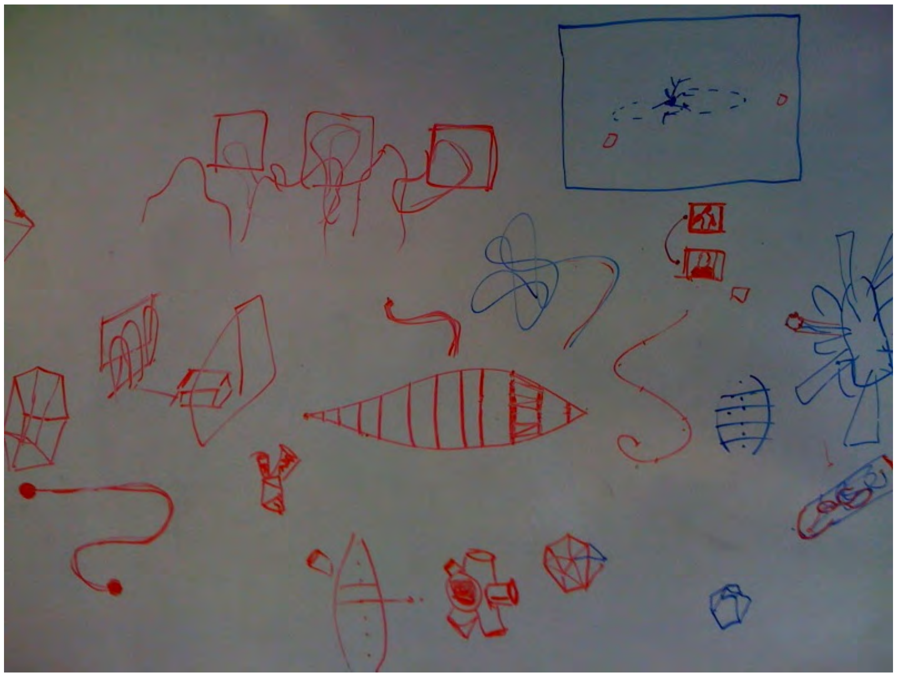


Figure 4.1 - Early brainstrom conducted on a white board.

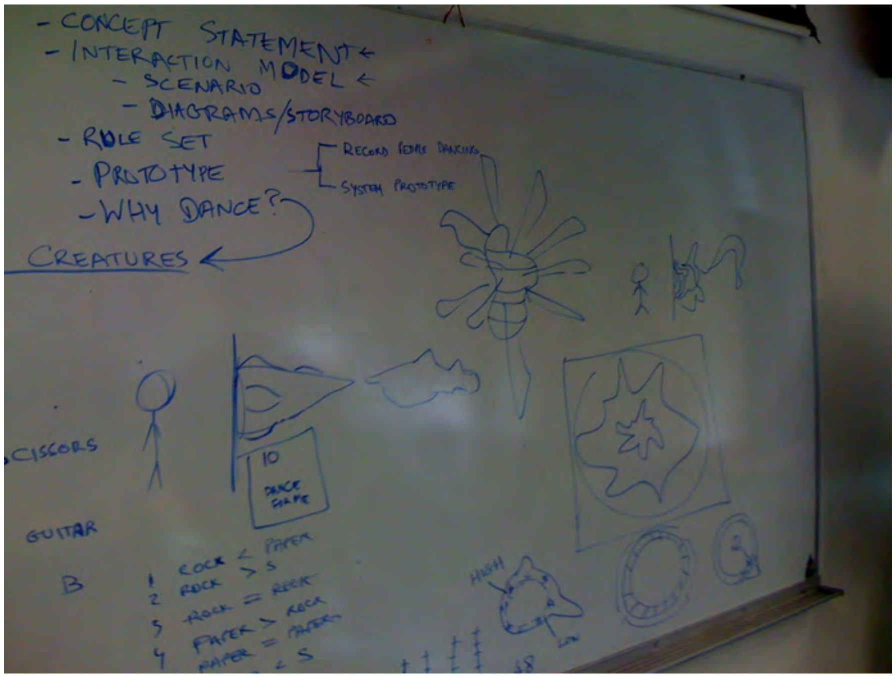


Figure 4.2 - Early brainstrom conducted on a white board.

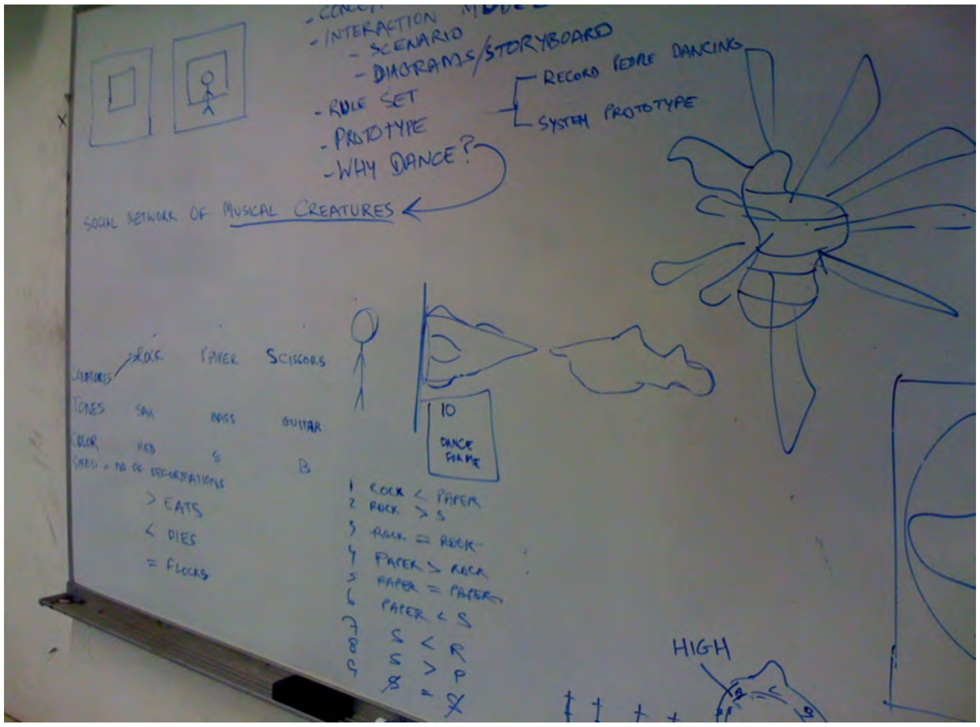


Figure 4.3 - Early brainstrom conducted on a white board.

ii. Role Prototypes

Behaviors

The first step to building a virtual environment with creatures was the need for these creatures to exhibit certain behaviors. The question we were trying to answer here was how would these creatures interact with each other, the environment and the user? We analyzed the simple rule set of the Rock, Paper, Scissors game and chose this to build on. The rule set was modified to suit our purposes and mapped to behavior of the three creatures. The rules resulted in two ways the creatures respond to each other - like creatures flock together and creatures higher up in the chain chase the creatures lower down in the chain. User interaction, or disturbance, results in altering the direction of movement of the creature and it's aural property. This is explained in more detail under the methodology section. The simple set of rules was easily understood by users and worked very well.

Storyboards/Interaction model

These mockups have the same impetus, intention and underlying interaction as the current project, so we will only highlight the differences.

Our first mock ups can be seen in Figures 4.4 - 4.8. While in playback mode and all the creatures are on the screen at once, users could influence them by texting certain messages to a specified phone number (Figure 4.4). During the "select your creature" scene (Figure 4.5), we chose to give the user the ability to pick the color of the worm, which would also determine the creature type. Next the user enters the record scene, which was composed of zones that varied in area and color, based on arbitrary decisions. The zones did not form a complete circle either (Figure 4.6). In this scene the user makes their creature very similar to our current version of the record scene, activating zones that enables notes and creates extrusions (Figure 4.7). The claim your creature scene (Figure 4.8) in this version relied on SMS to name their creature and attached the text message to the creature. As opposed to the current method of taking a picture.

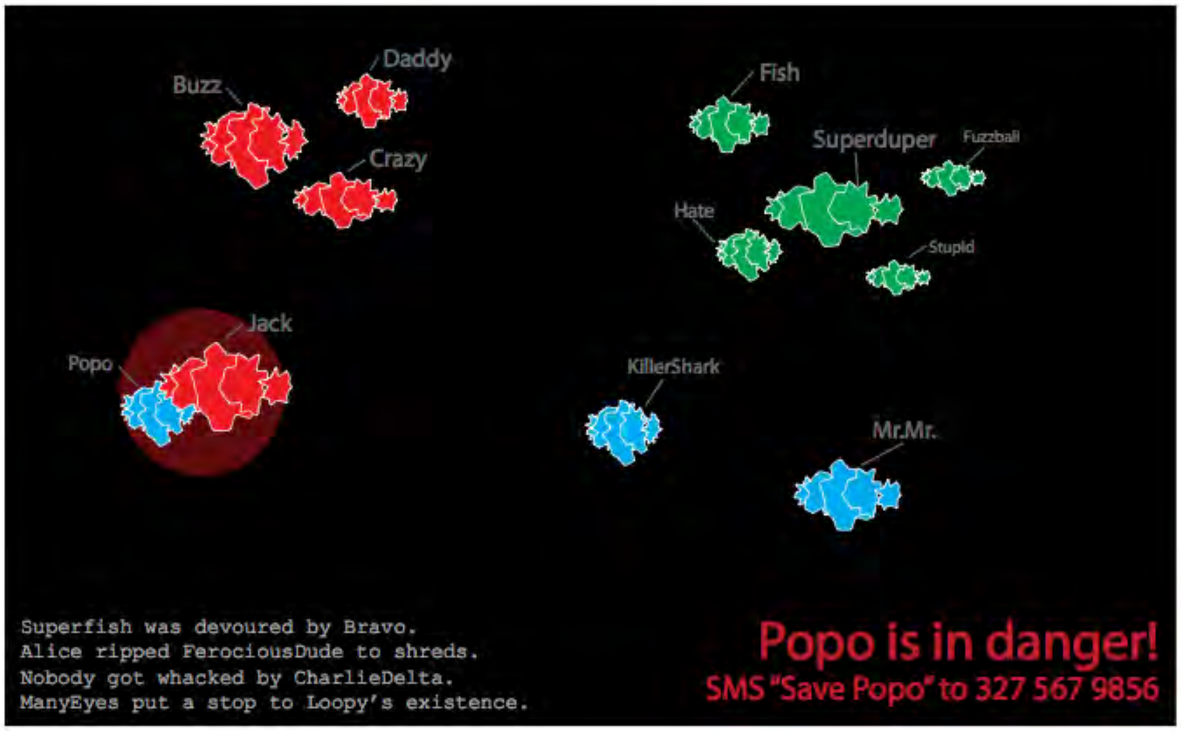

Daddy
Buzz
Crazy
Fish
Superduper
Fuzzball
Hate
Stupid
Jack
Popo
KillerShark
Mr.Mr.
Superfish was devoured by Bravo.
Alice ripped FerociousDude to shreds.
Nobody got whacked by CharlieDelta.
ManyEyes put a stop to Loopy's existence.
Popo is in danger!
SMS "Save Popo" to 327 567 9856

Figure 4.4 - Mock up of all the creatures roaming around in the virtual environment. Users could send commands to the creatures via SMS.

Figure 4.5 - Early mock up of "select creature scene," based on pre-determined colors.

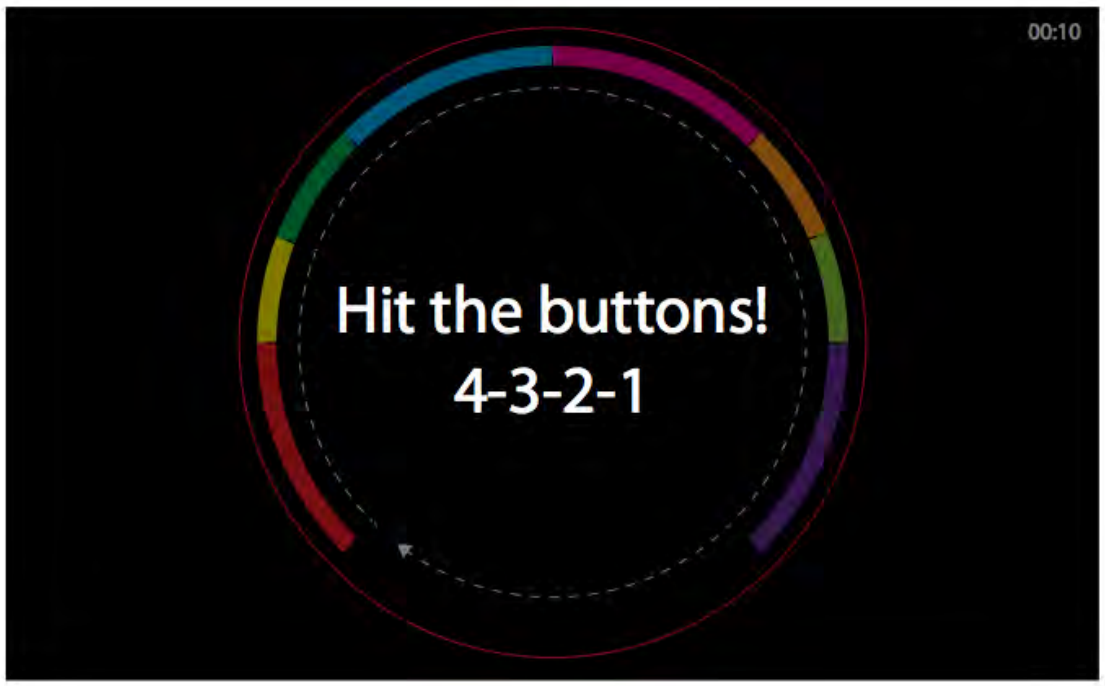


Figure 4.6 - Mock up of "record creature scene." The "hit zones" were given irregular sizes and colors and did not form a complete circle.

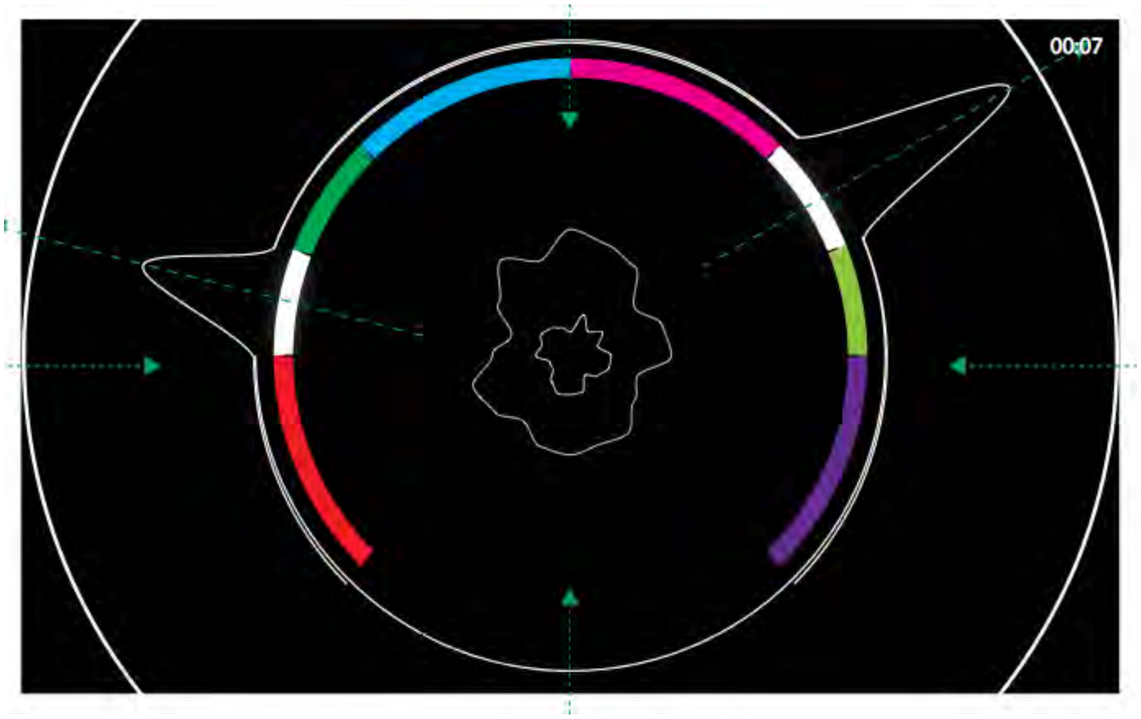


Figure 4.7 - Mock up of the "record creature scene" while a user is hitting two of the "hit zones" the thin line being pushed from the center to represent the extrusions.

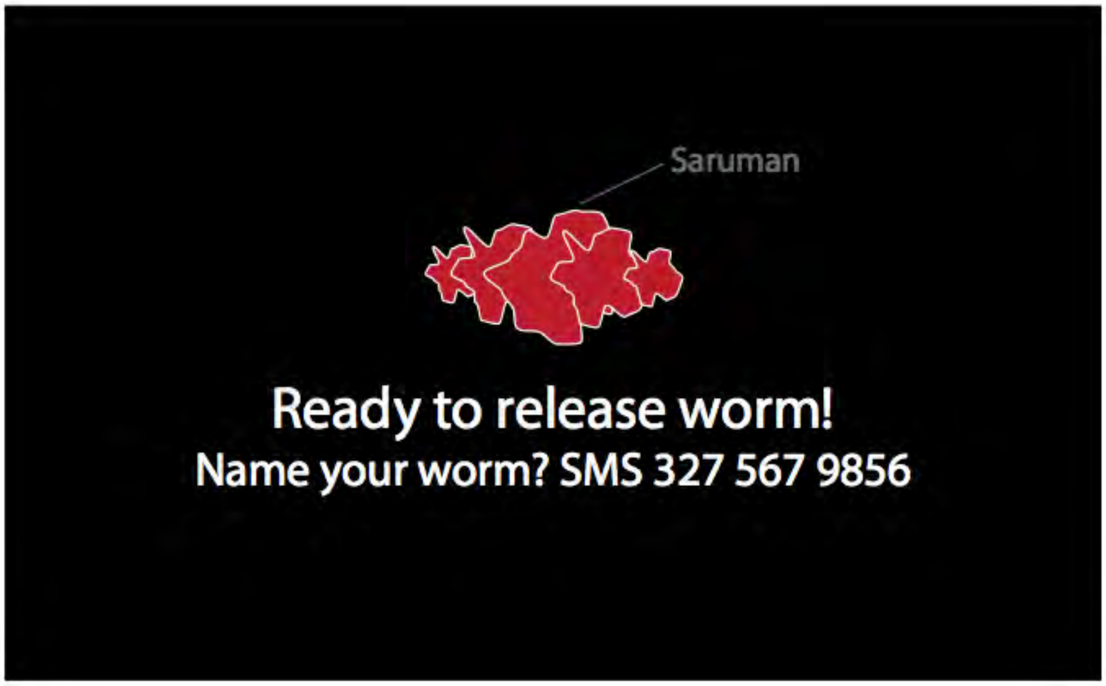


Figure 4.8 - Mock up of the "creature completed scene." Users could use their cellular phones to name their creature via SMS.

iii. Look and Feel

Mood boards

While trying to determine the look and feel of the creatures, we put together a mood board, to inspire and influence design, which can be seen below in Figure 4.4.

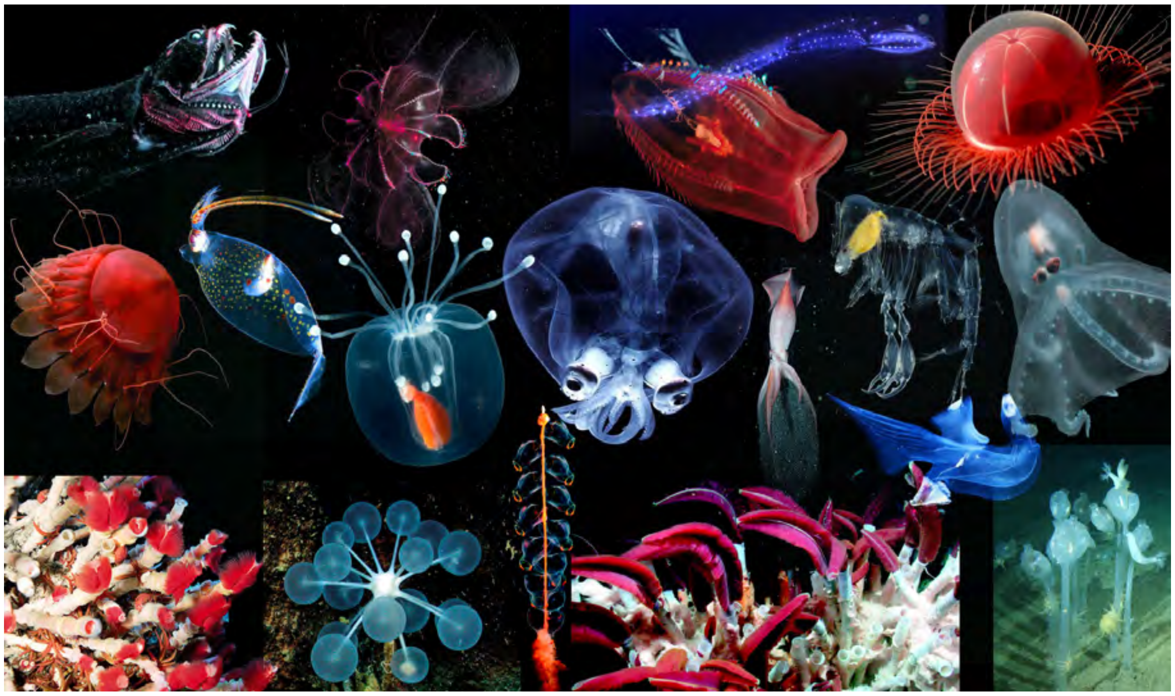


Figure 4.9 - Mood board for visual inspiration.

Creatures - 3D Models

What should the creatures look like? This is was the toughest challenge for us throughout the process. Once we finally arrived at an interaction model, the next step was to tie this into how the creatures look and what visual features would differentiate each creature from the other. The most common feature present in all the underwater creatures we drew inspiration from was the presence of tentacles, tendrils or something similar, present on the body of the creature. This led us to choose this as the similar feature that creatures in our system would have. The tentacles/tendril fit well with our interaction model as well and can be more appropriately termed extrusions from the creatures body. The interaction model is discussed in more detail under methodology. The extrusions needed to be differentiated to give each creature type an identity which led to three different styles - spiky, rectangular and more tendril-like. Simple 3D models of the creatures were produced in Maya (Figures 4.10 - 4.12) to help establish the look and feel and direction for visual programming.

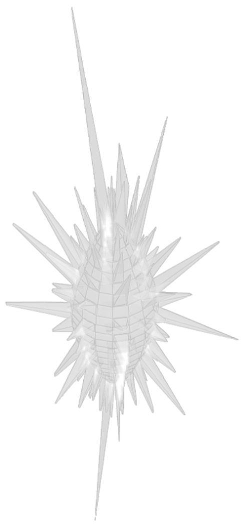


Figure 4.10 - Early Spiky creature rendering done in Maya.

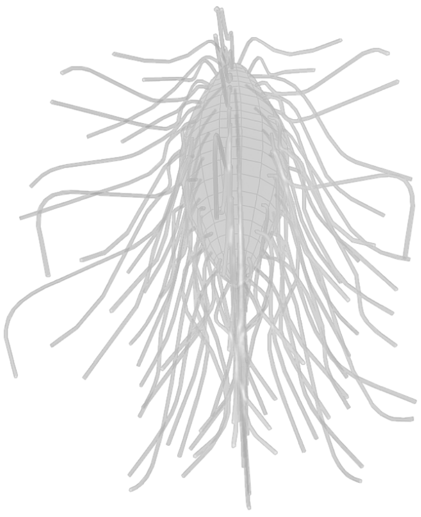


Figure 4.11 - Early Slinky creature rendering done in Maya.

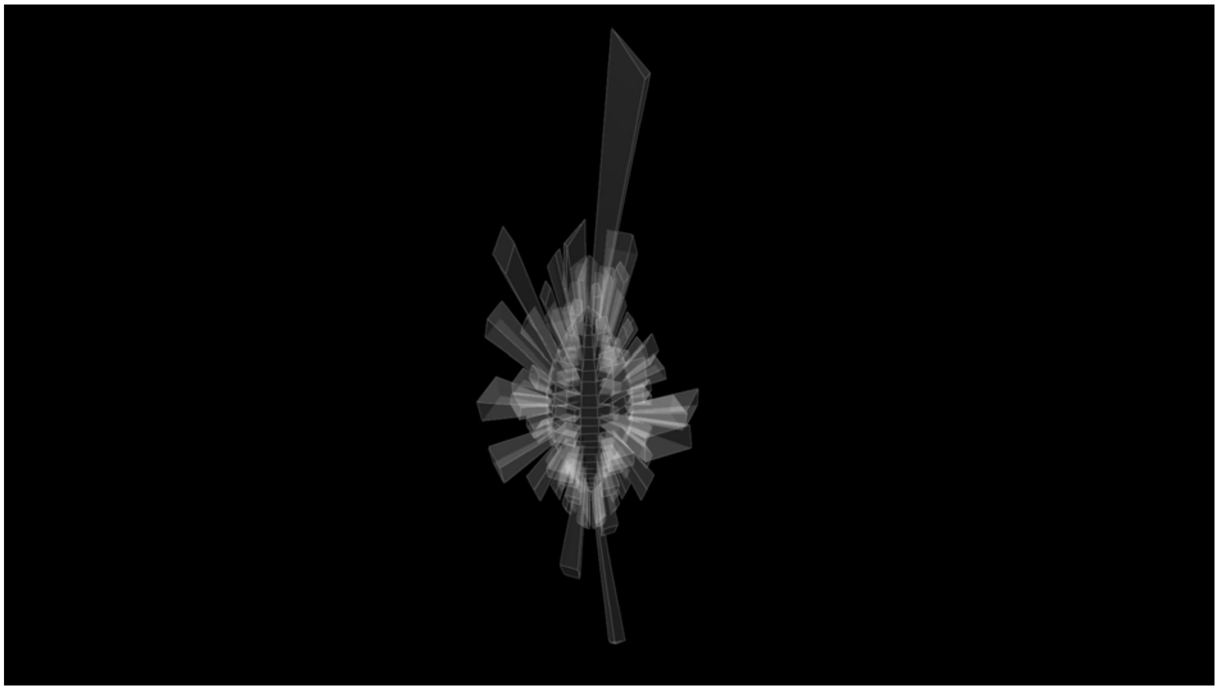
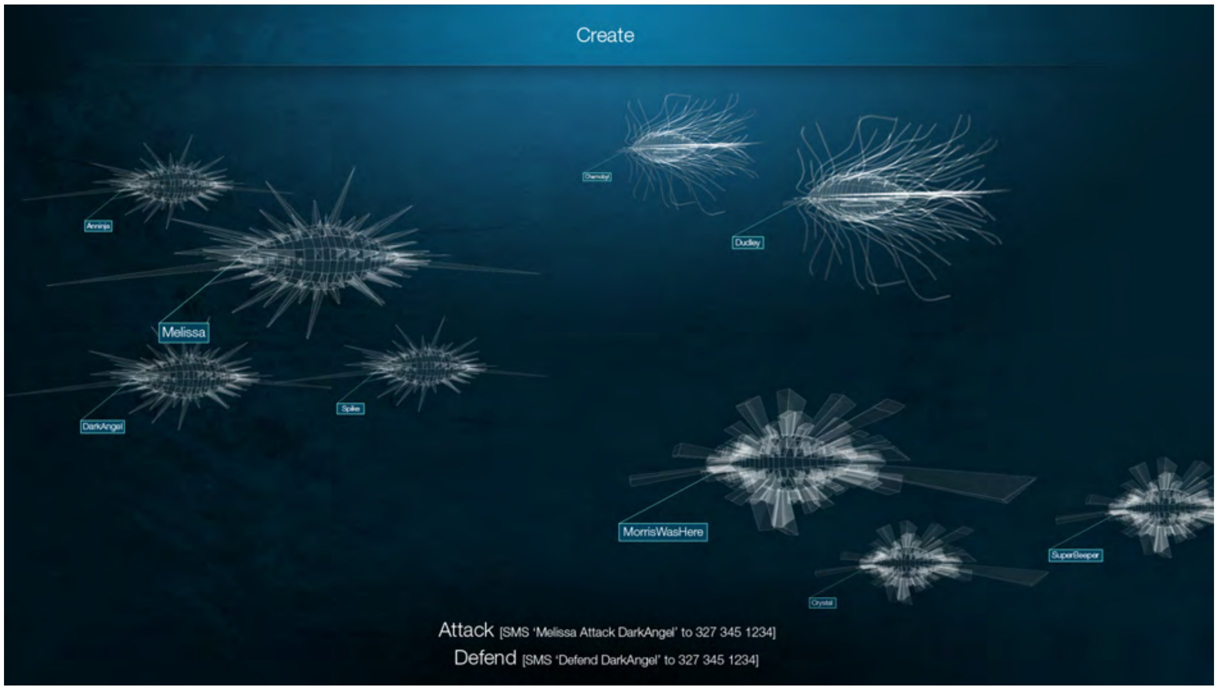


Figure 4.12 - Early Rectangles creature rendering done in Maya.

Mockups

The following mock ups (Figures 4.13 - 4.17) were created to find a more visually pleasing aesthetic and determine the main call to action, which turned out to be the create button. This is evident at the top of each board is text that prompts the user to do something like "Create" or "Deploy," for example. The user could still control the creatures via SMS sent from their phone, as evident in Figure 4.13. During this round of designs, we decided that the user should select the type of creature in the "select creature type" scene, shown in Figure 4.14. We chose this since selecting the color did not reveal the creature body type until the creating process was complete. We felt that the creature body type was more compelling than selecting between three pre-determined colors. The record scene was also limited to two arbitrary colors. The hit zones were each given an equal area, and formed a complete circle (Figure 4.15). The thin line that represents the extrusions forms shapes relevant to the creature type, Slinky is flowing, Spiky is pointed and Rectangles is geometric. A mock up of a Rectangles creature being created is shown in Figure 4.16. The claim creature scene (Figure 4.17), still relied on SMS to name the creature.



Attack [SMS 'Melissa Attack DarkAngel' to 327 345 1234]
Defend [SMS 'Defend DarkAngel' to 327 345 1234]

Figure 4.13 - Mock up displaying all the creatures in the environment. The create button was added and the top navigation was implemented in the other scenes as well. Users could control creature's actions via SMS.

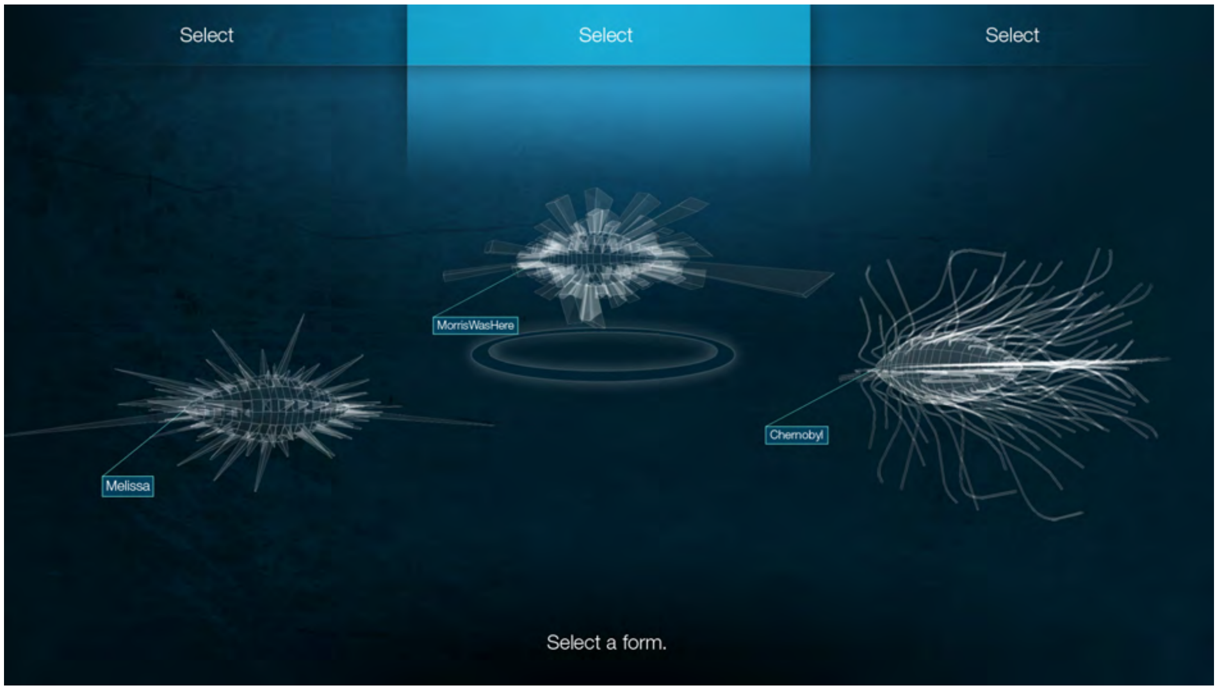


Figure 4.14 - Mock up of the "select creature scene," creature types now being displayed, instead of only color.

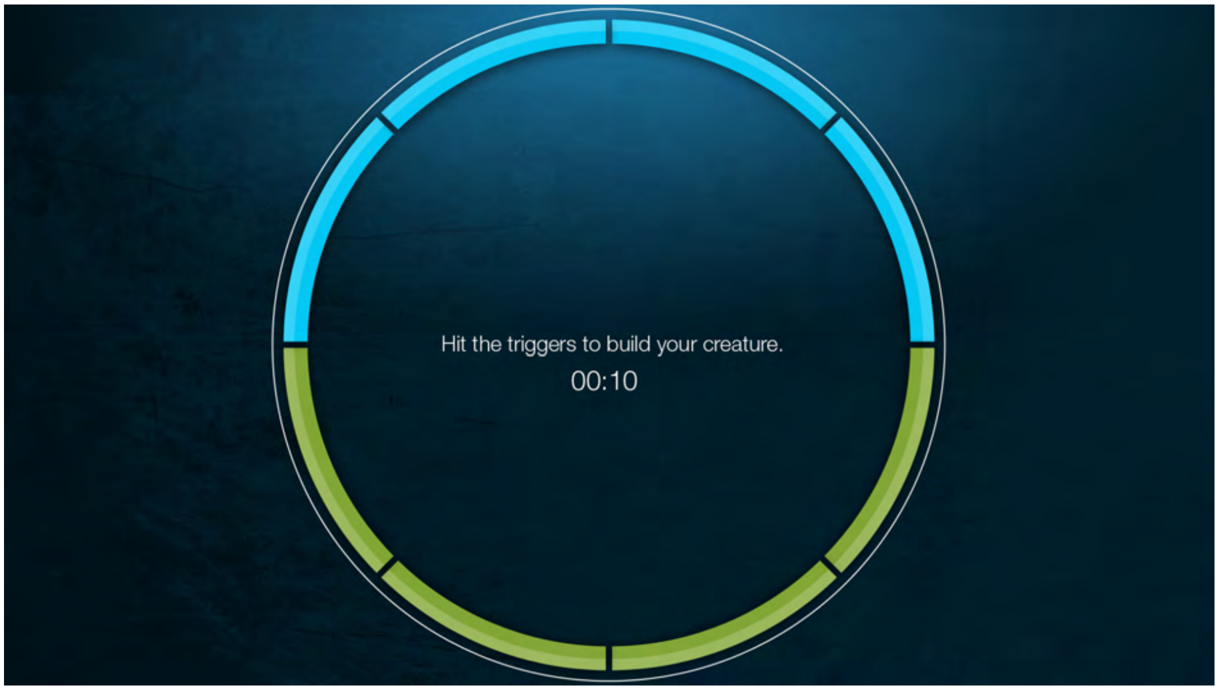


Figure 4.15 - Mock up of the "record creature scene." Each "hit zone" is now equal size and forms a complete circle, and given one of two arbitrary colors.

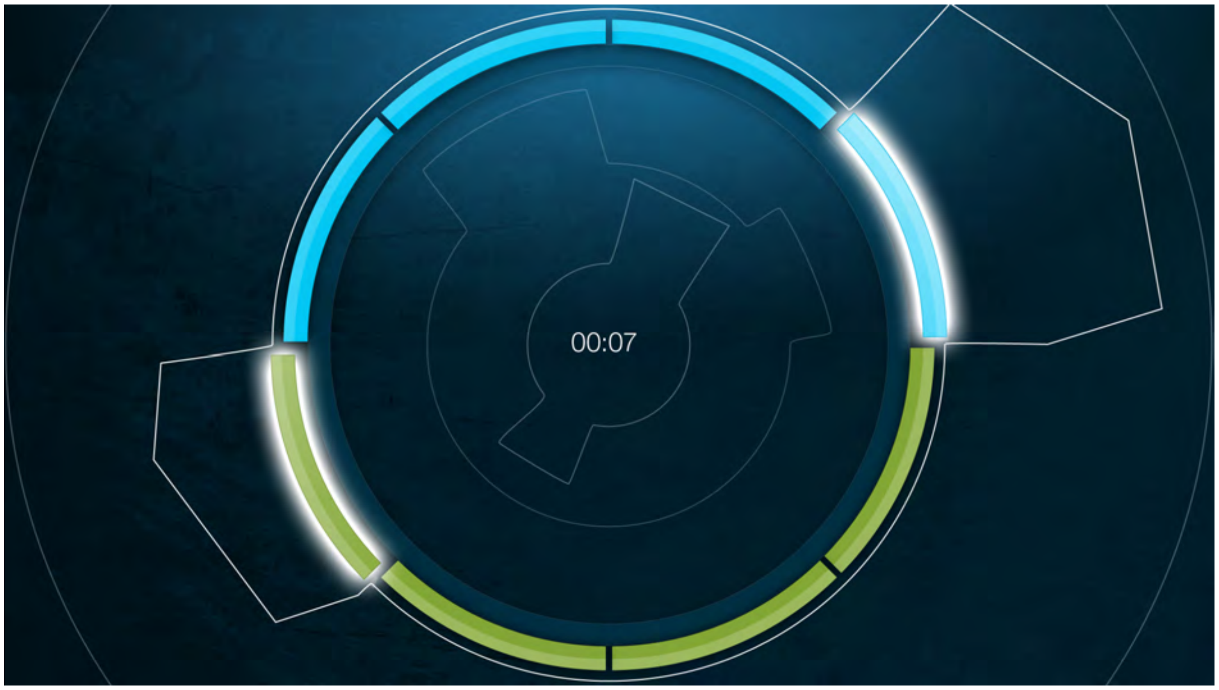


Figure 4.16 - Mock up of "record creature scene," while some of the extrusions have been created. The thin line that represents the extrusion forms shapes relevant to the creature type, Slinky is flowing, Spiky is pointed and Rectangles is geometric and is the type represented in this mock up.

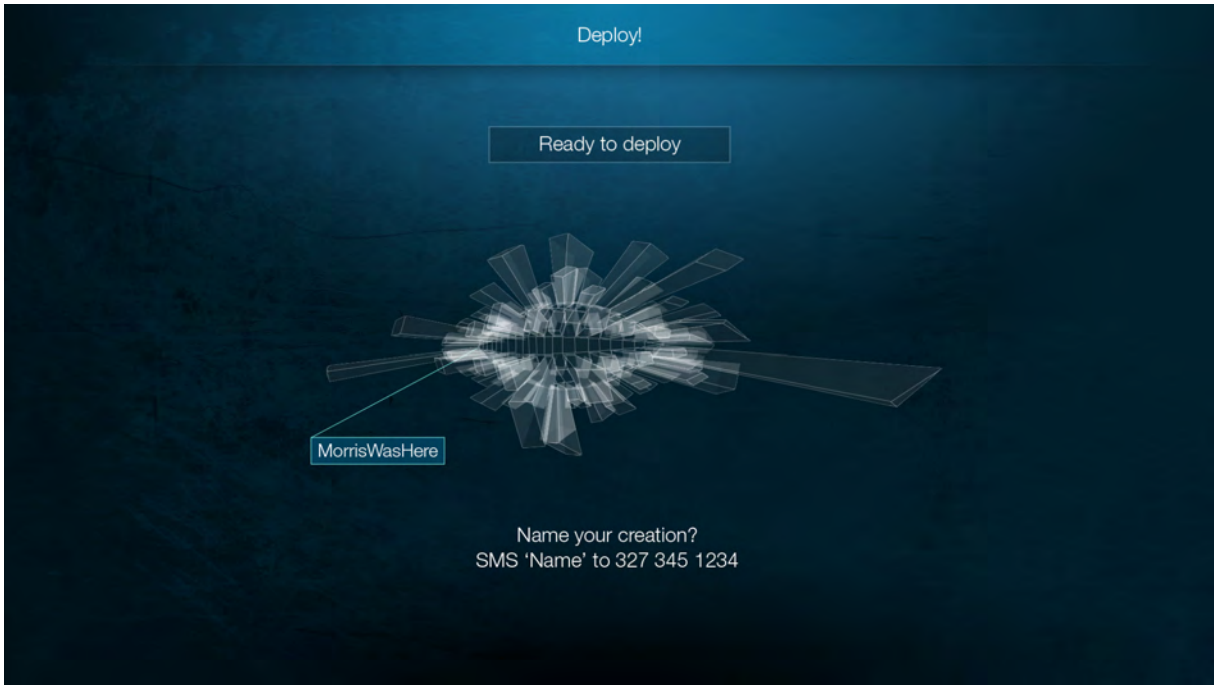


Figure 4.17 - The "completed creature scene." The creatures are still named via SMS from the user's phone.

The final round of visual mock ups are the current version (Figure 4.18). The main goal of this round of designs was to eliminate the need for anything external, that would create a barrier of interaction. As of the last round of designs, this was the cellular phone. Users would have to be carrying a cellular phone, have a SMS enabled phone, be willing to SMS the project number, type the correct message and still be engaged with the installation after drawing their attention away from projection. We also felt that this is impersonal, as the computer would display all of the names in the same font and font size, in a uniform manner, which would not be unique like a handwritten signature. We started thinking of ways that we could have the user 'tag' their creature, and since we were already tracking motion, contemplated the user signing their creature through their movement. We thought this may turn out messy and would not be easy to identify. Then we thought of the camera, and what is more personal than a photograph of yourself. This is a familiar domain with the user and the process is completed quickly. We chose to utilize black and white photographs to maintain the overall aesthetic and keep the focus on the creatures and not the thumbnails of people in bright clothes.

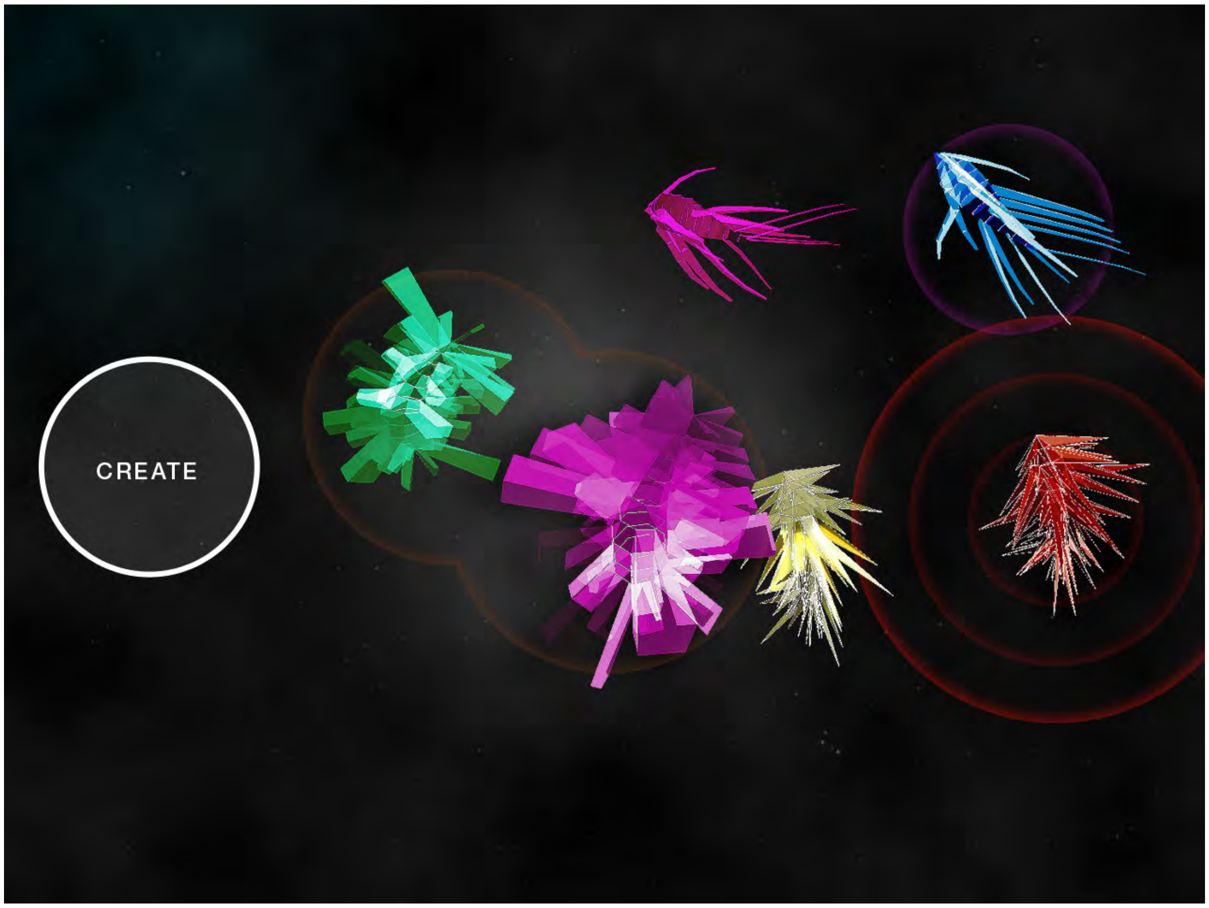


Figure 4.18 - Latest mock up of all creatures in the environment.

To further simplify the interaction, we eliminated the phone all together and made the interaction with the creatures in playback mode with motion. This cut down one more level of interaction and made the project cohesive with the interaction solely through body movement. This would also allow users to explore and become acquainted with the interaction model, fostering better informed decisions in the more complex interaction of creating a creature.

iv. Implementation

Worm experiments

The Fixed Up Method, shown in Figure 4.19. was initially used for determining the geometry for the body of the worms. The method worked well, except for one major problem, when the spine vector aligned or came close to parallel with the up vector, twisting occurred, as illustrated in Figure 4.20. I posted this issue to the OpenFrameworks forums (http://www.openframeworks.cc/forum/viewtopic.php?f=8&t=3383). Thanks to the forum users Edvin and Damian, we figured it out.

We set the initial point of the worm to influence the up vector of the worm for the rest of the tube that used parallel transport. Based on Craig Reynolds [18] information, set the approximate up vector to the up vector of the previous frame and the forward vector to the direction normal. Then set the side vector to the cross of the forward and approximate up vector. Then set the new up to the cross of the forward and the side vector.

Based on the awesome Game Programming Gems 2 [2] book recommended by Edvin, performed the Parallel Transport Frame Method on the rest of the body points to extrude along the path and it solved the problem, as seen in the code example Figure 4.21.

```
ofxVec3f cross = dirNormal.perpendiculared(up);
cross.rotateRad(angle, dirNormal);
cross *= dist.length();
```

Figure 4.19 - Fixed Up Method Code performed for each spine point.

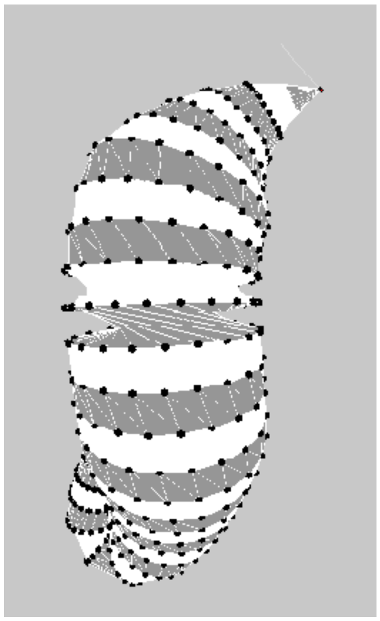


Figure 4.20 - Twisting around spine using the Fixed Up Method

```
if (i == 0) {
   // dirNormal is a normalized vector between spine points //
   // Reynolds Method [url]http://red3d.com/cwr/steer/gdc99/index.html[/url]
   ofxVec3f new_forward(dirNormal.x, dirNormal.y, dirNormal.z);
   new_forward.normalize();
   ofxVec3f approx_up(-spine[i].orientation.up.x, -spine[i].orientation.up.y, -
spine[i].orientation.up.z);
   approx_up.normalize();
   ofxVec3f new_side = new_forward.crossed(approx_up);
   ofxVec3f new_up = new_forward.crossed(new_side);
   new_up.normalize(); // just in case

   spine[i].orientation.up.set(new_up.x, new_up.y, new_up.z);
   spine[i].orientation.forward.set(new_forward.x, new_forward.y, new_forward.z);
   spine[i].orientation.side.set(new_side.x, new_side.y, new_side.z);

   ////////////////////////////////////////////////////////////////////////////
}  else if (i < numBodyPts) {
   // Game Programming Jems 2 Approach ///////////////////////////////
   ofxVec3f T1(spine[i-1].loc.x, spine[i-1].loc.y, spine[i-1].loc.z);
```

```
  ofxVec3f T2(spine[i].loc.x, spine[i].loc.y, spine[i].loc.z);
  ofxVec3f A = T1.crossed(T2);
  float alpha = T1.angleRad(T2);

  ofxVec3f new_forward(dirNormal.x, dirNormal.y, dirNormal.z);
  ofxVec3f approx_up(-spine[i-1].orientation.up.x, -spine[i-1].orientation.up.y, -spine[i-
1].orientation.up.z);
  approx_up.normalize();
  approx_up.rotateRad(alpha, A);
  ofxVec3f new_side = new_forward.crossed(approx_up);
  ofxVec3f new_up = new_forward.crossed(new_side);

  spine[i].orientation.up.set(new_up.x, new_up.y, new_up.z);
  spine[i].orientation.forward.set(new_forward.x, new_forward.y, new_forward.z);
  spine[i].orientation.side.set(new_side.x, new_side.y, new_side.z);
}
```
Figure 4.21 - Parallel Transport Frame Method C++ code inside the body points loop.

Audio experiments

Reaktor

In our first several prototypes, we explored possible creature structures and their corresponding behaviors and movements. In addition, we linked attributes such as position and 'breath' to sound generation. We created a creature referred to as the squid (Figure 4.22) because his movements are similar to that of a squid. The squid breathes in and out, displayed by his body expanding and contracting. The squid propels himself around the screen by breathing out or exhaling quickly. This protoype generates sound utilizing the squid's breath percent, which is the proportion of the squid's breath to the total breath. This variable is sent to Reaktor via OSC (Open Sound Control) and changes the sound generated.

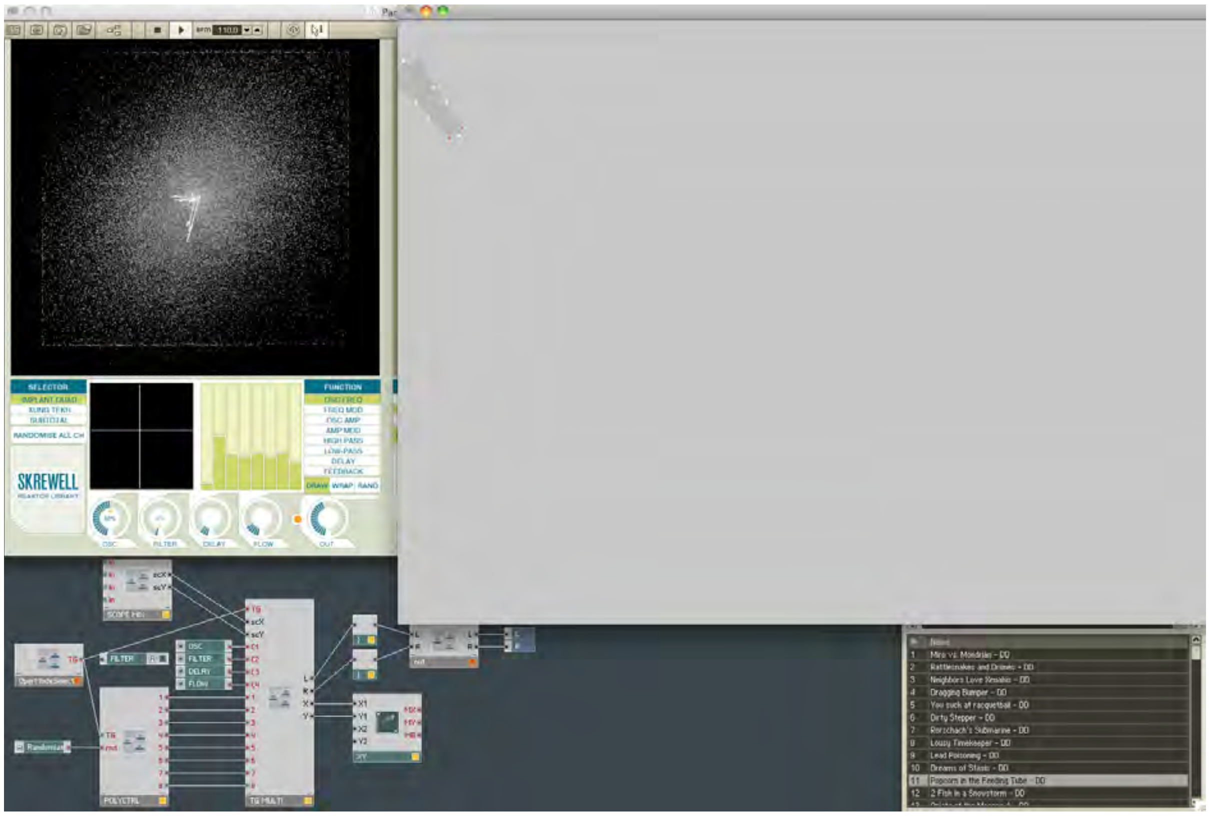


Figure 4.22 - Squid creature on right controlling audio output in Reaktor (visual displayed on left)

In the next iteration we constructed another creature which we refer to as the tick, due to the tiny size and quick kicking motion (Figure 4.23). In this iteration, the creatures follow the mouse. Since the user will most likely be moving, this helped us learn how the creatures react to moving targets. The tick also generates sound in Reaktor, based on the kicking motion. If the leg is fully extended, a bass drum sound is triggered.

We also incorporated sliders in this experiment (Figure 4.24). The sliders enabled the manipulation of the creature in real time and we could quickly see and hear the results. Although this was successful, feedback indicated that it might not be the best form of input in the final form.

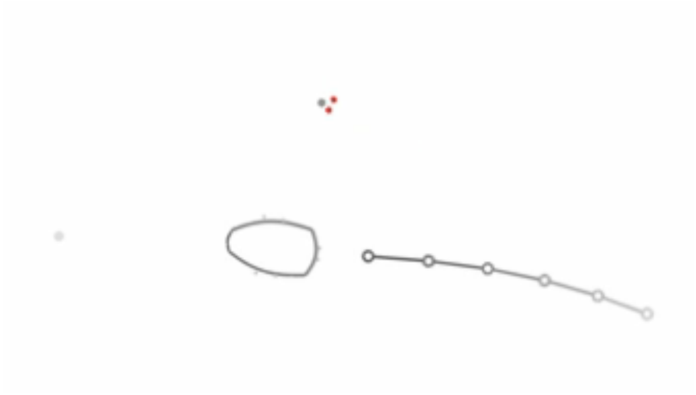
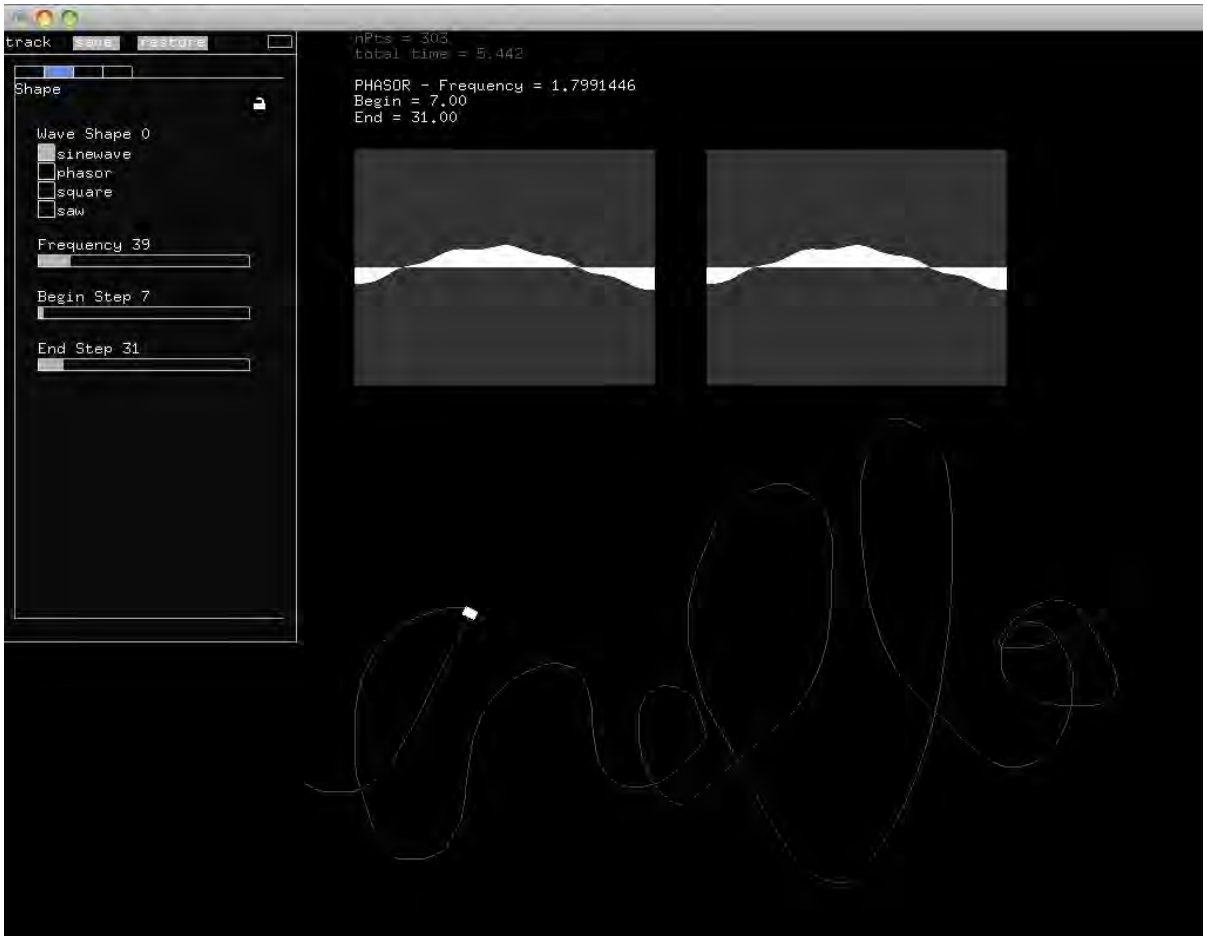


Figure 4.23 - Tick (three circles, one grey and two red) and Squid creature prototypes, organic movement through 2D Perlin Noise

Figure 4.24 - Kicking of legs controlled by slider, which generates audio from Reaktor

Maximilian

After all the experiments with Reaktor we arrived at the conclusion that we needed more granular and dynamic control over how the sound was created. This led us to experiment with audio synthesis programming. The first experiment was conducted with Maximilian, a cross platform C++ audio synthesis library [5]. The plugin for this library was used with the openFrameworks toolkit to create an audio synthesis tool that explored the flexibility and possibilities available via Maximilian (Figure 4.25). The library provides classes for standard waveforms, sample playback, envelopes, filters with resonance and delay lines along with equal power stereo, quadraphonic and 8-channel ambisonic support. This library did not prove to be very useful due to a limited amount of sonic possibilities. We finally used SuperCollider for audio synthesis which is described in more detail under the methodlogy section.



Wave Shape 0
[✓ sinewave]
phasor
square
saw

Frequency 39

Begin Step 7

End Step 31

```
nPts = 303
total time = 5.442

PHASOR - Frequency = 1.7991446
Begin = 7.00
End = 31.00
```

Figure 4.25 - Audio synthesis tool created with maximilian

Interaction experiments

How does the participant create a creature? This was one of the most challenging questions. We wanted to make this possible without the interaction being too complex to understand or too involved to participate in.

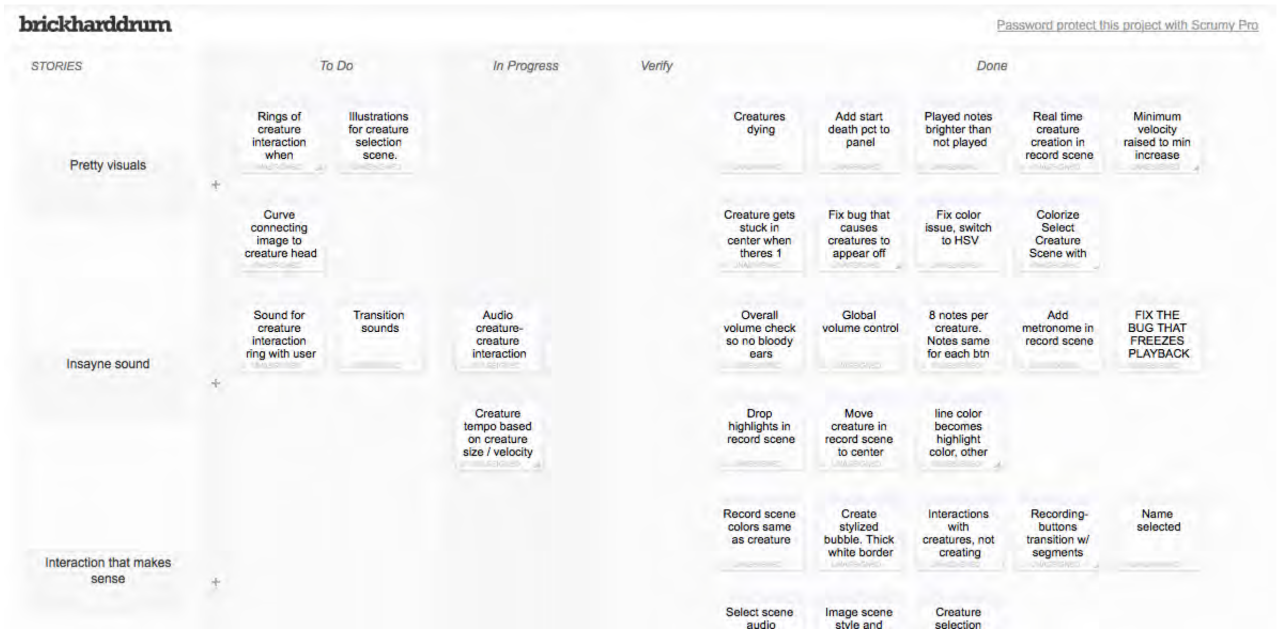
One of our earlier interaction experiments was built around a drawing interface. Once a user is detected, their picture is instantly taken and placed in the top right corner of the screen. They enter a recording mode that is evident by a line on the screen that traces their motion throughout the physical space via a camera. The user is also aware that their movement is being recorded by a small dot moving about the screen and their motion history being faded out over time, giving a smoky effect of their camera image.

This interaction is determined to be complete when a user stops moving or the set time limit is exceeded, approximately 10 seconds.

After completion, the line that was created by the user's motion is converted to 3D and animated to attaching itself to a section on the main worm-like structure. This serves as a time-line for altering the overall sound and time replay is evident as a glowing orb revolves around the line in 3D space. Every time that a user creates one of these "recordings," which can be interpreted as data, the "recordings" attach themselves to a segment of the worm, moving the previous ones down a segment. Once a "recording" is pushed off of the end of the tail of the worm, then it becomes part of the 3D environment, like an artifact. This interaction was soon discarded based on feedback. The next iteration was a version of the final implementation that is discussed under Storyboards/Interaction model and in great detail under the methodology section.

Scrum

We used Scrumy [21] to help keep track of tasks we needed to accomplish throughout the production and development process. Scrumy is project management tool loosely based on Scrum [13] - an iterative framework for agile software development and project management. This tool helped us not only approach the project with an organized approach but gave us a sense of accomplishment as we moved tasks over to the "Done" column. A sample of our scrum can be seen below in Figure 4.26.



brickharddrum

Password protect this project with Scrumy Pro

STORIES	To Do		In Progress	Verify	Done				
Pretty visuals	Rings of creature interaction when	Illustrations for creature selection scene.			Creatures dying	Add start death pct to panel	Played notes brighter than not played	Real time creature creation in record scene	Minimum velocity raised to min increase
	Curve connecting image to creature head				Creature gets stuck in center when theres 1	Fix bug that causes creatures to appear off	Fix color issue, switch to HSV	Colorize Select Creature Scene with	
Insayne sound	Sound for creature interaction ring with user	Transition sounds	Audio creature-creature interaction		Overall volume check so no bloody ears	Global volume control	8 notes per creature. Notes same for each btn	Add metronome in record scene	FIX THE BUG THAT FREEZES PLAYBACK
			Creature tempo based on creature size / velocity		Drop highlights in record scene	Move creature in record scene to center	line color becomes highlight color, other		
Interaction that makes sense					Record scene colors same as creature	Create stylized bubble. Thick white border	Interactions with creatures, not creating	Recording-buttons transition w/ segments	Name selected
					Select scene audio	Image scene style and	Creature selection		

Figure 4.26 - Scrumy.com web project

3. FUTURE STEPS & PROSPECTS

a. Revisions

We would like to implement a few revisions for future iterations. While observing people interact with this project, we noticed that most people selected the middle creature, Spiky, during the "Select Creature Scene." We could not determine if this is the most popular creature because it is in the center or because people prefer its aesthetic. For this reason, we would like to position the creatures in random order, for a non-biased selection process.

Currently, once a creature dies, it is no longer, we are not storing the creature, nor the user image. We would like to store the creature's attributes so that it could be recreated at any time. The user's image and creation time would also be attached to the stored creature for identification. We are not sure how / why to apply, visualize or recreate the creatures, but saving the data would preserve the creatures, and thus user's interactions.

b. Web distribution

Currently we are developing a documentation website for the project that can be found at social.sqncr.com. Once the website is complete, we plan to promote it through social media sites like Facebook, Twitter, Vimeo and the like. We also plan to submit to blogs that share and promote creative work. These sites include, but are not limited to:

- http://www.informationisbeautiful.net
- http://infosthetics.com
- http://www.psfk.com
- http://www.creativeapplications.net
- http://createdigitalmotion.com
- http://visualcomplexity.com

We hope to create an awareness of our project in the online community and then install the project in as many festivals and conferences that we can possibly get accepted to. We would also like to release the software under an Open Source Initiative - MIT License in order to share it with the larger community and people who would like to build similar installations.

c. Conferences

We are going to demo social sqncr at the Humanities + Digital Visual Interpretations Conference (http://hyperstudio.mit.edu/events/) at MIT on May 22. We will use this venue to promote our project and gather constructive criticism and feedback. Based on the feedback received, we may decide to change or add features to the project. Some of the additional festivals that we would like to submit include:

- Piksel 2010 (http://piksel.no/ocs/index.php/piksel/piksel10)
- Ars Electronica 2011 (http://www.aec.at/about_about_en.php)
- Prixars 2011 (http://www.aec.at/prix_about_en.php)
- SIGGRAPH 2011 (http://www.siggraph.org/)

- SIGCHI 2011 (http://www.chi2011.org/)

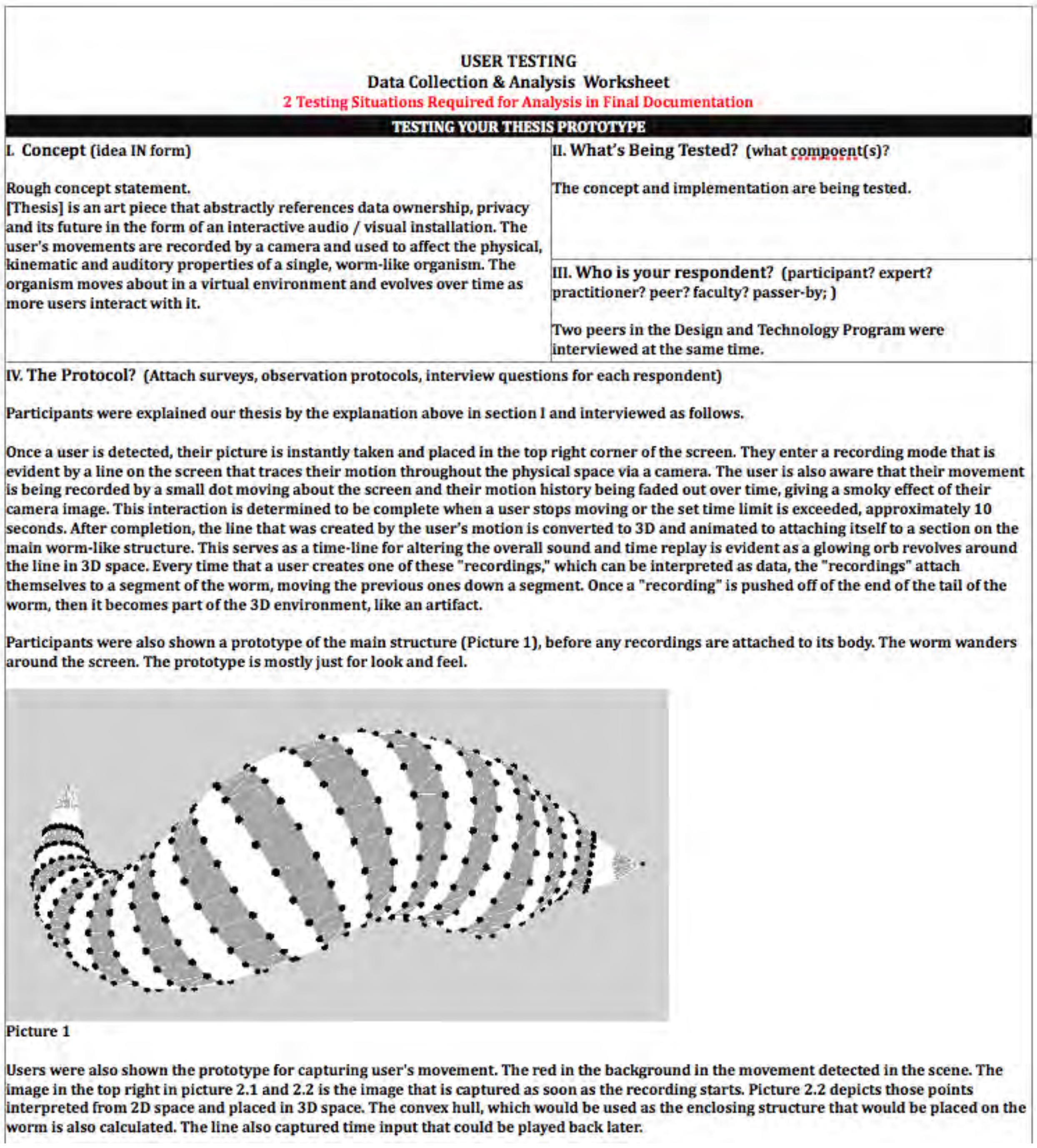
Since this project is an installation, we plan on documenting it in use as much as possible, as that is how the project will be remembered by those who could not experience it physically.

References

1. Akten, M., *Glastonbury Pi*. 2008. http://www.msavisuals.com/glastonbury_pi
2. DeLoura, M., *The Parallel Transport Frame. Game Programming Gems 2.* pp. 215 - 219. Charles River Media, Inc. Hingham, Massachusetts. 2001.
3. FAO Schwarz, *Large Piano Installation at Fifth Avenue in New York*. http://en.wikipedia.org/wiki/FAO_Schwarz
4. Galanter, P., BA, MFA, Interactive Telecommunications Program, New York University, New York, USA.
5. Grierson, M., Maximilian: A Cross Platform C++ Audio Synthesis Library for Artists Learning to Program, Goldsmiths Department of Computing, University of London. January 2010.
6. Houde, S. and Hill, C. What do prototypes prototype. Handbook of Human Computer Interaction, 1997.
7. Krueger, Myron. Video Place. 1975. http://www.mindatplay.co.uk/videoplace.html
8. Levin, Golan. Audio Visual Environment Suite. 1998 - 2000. http://www.flong.com/projects/aves/
9. Lieberman, Zachary. "Co-Founder of OpenFrameworks." http://www.thesystemis.com/.
10. Lieberman, Zach. Watson, Theodore. and Castro, Arturo. OpenFrameworks. C++ Library. http://openframeworks.cc/.
11. Limiter. http://en.wikipedia.org/wiki/Limiter
12. Miller, J. G., Living Systems, University Press of Colorado (May 1995).
13. Mountain Goat Software, Introduction to Scrum - An Agile Process, http://www.mountaingoatsoftware.com/topics/scrum
14. Open Graphics Library. C++ Library. Maintained by Gold Standard Group. http://www.opengl.org/.
15. Open Sound Control. Computer Protocol. UC Berkley. http://opensoundcontrol.org/.
16. Open Source Computer Vision. C++ Library. http://opencv.willowgarage.com/wiki/.

17. Perlin, Ken. "Java Reference Implementation of Improved Noise." http://mrl.nyu.edu/~perlin/noise/. 2002.
18. Reynolds, Craig. Flocks, Herds and Schools: A Distributed Behavioral Model. ACM SIGGRAPH '87 Conference Proceedings, Anaheim, California, July 1987
19. Sommerer, Christa. Mignonneau, Laurent. GENMA - Genetic Manipulator. 1996-97. http://www.interface.ufg.ac.at/christa-laurent/WORKS/FRAMES/FrameSet.html
20. SuperCollider. Software. http://www.audiosynth.com/.
21. Scrumy. http://scrumy.com
22. Twiddla. http://www.twiddla.com/
23. Watson, Theodore. Emily Gobeille. Terrarium. 2008. http://www.theowatson.com/site_docs/work.php?id=44.

Appendix A: User Testing Worksheets



USER TESTING
Data Collection & Analysis Worksheet
2 Testing Situations Required for Analysis in Final Documentation

TESTING YOUR THESIS PROTOTYPE

I. Concept (idea IN form)

Rough concept statement.
[Thesis] is an art piece that abstractly references data ownership, privacy and its future in the form of an interactive audio / visual installation. The user's movements are recorded by a camera and used to affect the physical, kinematic and auditory properties of a single, worm-like organism. The organism moves about in a virtual environment and evolves over time as more users interact with it.

II. What's Being Tested? (what compoent(s)?

The concept and implementation are being tested.

III. Who is your respondent? (participant? expert? practitioner? peer? faculty? passer-by;)

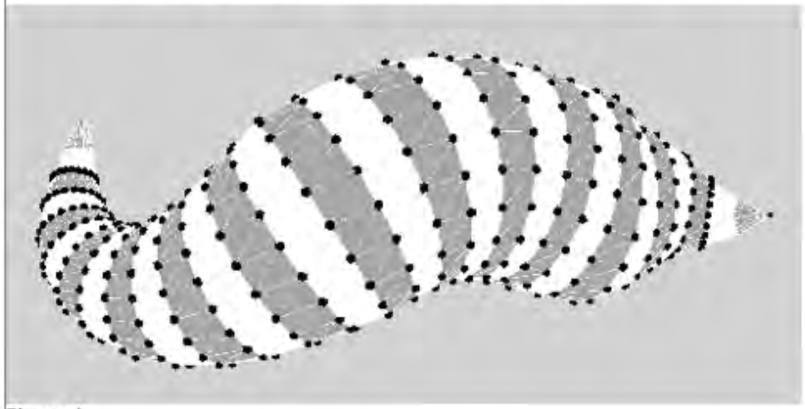
Two peers in the Design and Technology Program were interviewed at the same time.

IV. The Protocol? (Attach surveys, observation protocols, interview questions for each respondent)

Participants were explained our thesis by the explanation above in section I and interviewed as follows.

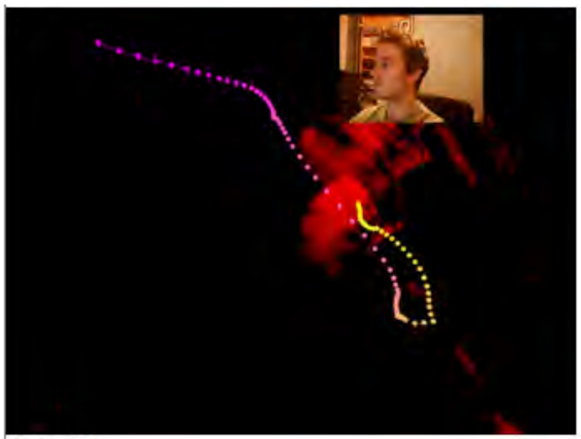
Once a user is detected, their picture is instantly taken and placed in the top right corner of the screen. They enter a recording mode that is evident by a line on the screen that traces their motion throughout the physical space via a camera. The user is also aware that their movement is being recorded by a small dot moving about the screen and their motion history being faded out over time, giving a smoky effect of their camera image. This interaction is determined to be complete when a user stops moving or the set time limit is exceeded, approximately 10 seconds. After completion, the line that was created by the user's motion is converted to 3D and animated to attaching itself to a section on the main worm-like structure. This serves as a time-line for altering the overall sound and time replay is evident as a glowing orb revolves around the line in 3D space. Every time that a user creates one of these "recordings," which can be interpreted as data, the "recordings" attach themselves to a segment of the worm, moving the previous ones down a segment. Once a "recording" is pushed off of the end of the tail of the worm, then it becomes part of the 3D environment, like an artifact.

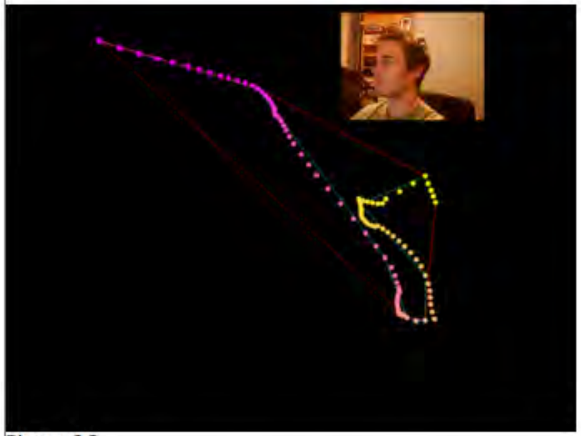
Participants were also shown a prototype of the main structure (Picture 1), before any recordings are attached to its body. The worm wanders around the screen. The prototype is mostly just for look and feel.



Picture 1

Users were also shown the prototype for capturing user's movement. The red in the background in the movement detected in the scene. The image in the top right in picture 2.1 and 2.2 is the image that is captured as soon as the recording starts. Picture 2.2 depicts those points interpreted from 2D space and placed in 3D space. The convex hull, which would be used as the enclosing structure that would be placed on the worm is also calculated. The line also captured time input that could be played back later.


Picture 2.1


Picture 2.2

This was an informal interview and responses derived questions and elaborations that lead to some very useful insight.

V. Feedback (Brief notes/list)

1. Does this convey a data network? Do you feel that the worm symbolizes the movement of data?
Based on everything said and looking at the worm prototype, a data network did not come to mind. However, upon explanation it made sense. However, it was an abstract interpretation, and an explanation is definitely needed. Not sure if users are expected to read something before viewing this, but it feels like it will be over the user's heads.

2. Will the users know to interact with the piece? Will they want to?
People may not know what to do and they just stand there, which will not result in any movement detection. They could make facial expressions, which are undetectable by motion detection. They wave, which is boring and uninteresting.
Some possible ways to combat these interactions is to present them with something ans ask them to describe it with a physical motion.
If you were angry, happy, etc. What would you look like / act like?
If you were Facebook, Flickr, etc. What would you look like / act like?
Describe this picture. [Show picture of popular person, recent event, trending topic etc].
Could ask people this for user testing and video their responses.

3. What is your response to the functionality of the worm?
People could make paintings and this gets mapped to tentacles. The mapping makes certain sounds.
One creature, many tentacles. How do the tentacles relate to each other? cf. Usman Haque's talk, going from reactivity to interactivity.
Or, each person adds their own creature: Their painting makes a shape that has certain qualities.
This gets put on the inside of the creature like DNA.
Their painting "connects"/"plugs in" to certain spots inside, which activate certain creature qualities.
This is how a screenful of creatures left by different people interact with each other.
Quartz Composer visually plugs bits together to make the final result.

4. Each person creating their own creature is much more personal and could relate more to networks, social networks in particular, any elaborations on that idea?
Yeah the creatures could have social characteristics themselves. You could implement the Rock, Paper, Scissors method.
It is an easy algorithm that could determine the behaviors of the creatures, although you may want to add more that just three types of creatures. "Rocks" could eat "Scissors" but be eaten by "Papers", and well, you know how to play. Once one eats the other, then it could inherit some of the properties of that creature.
Could the changes be less about total change (one eats the next) or blending (averaging of qualities quickly goes to mud)...
Changing one quality depending on what the two contenders are (see RPS above).
E.g.: duration of the sound, number of times it repeats, using sawtooth waveform instead of sine wave, the hue or saturation or brightness of the creature, its overall size, how its tentacles move, how many tentacles it has...
This is definitely like an eco system and you would have to find the right balance for each creature.

5. Any ideas on how the creature characteristics should be determined? Should they be auto-assigned? Chosen by the user?
Hmm. That is a tough question because if you allow the user to choose, then the system may result in only one type of creature, but it lets the user have more control. It also gives them more participation in the activity. Then they may come back from time to time to see if their creature is still alive. Which means that there needs to be a strong visual distinction of their creature among other, but still know which ones they are related to.
This could be done through color. Having three shades, for Rock, Paper, Scissors and then their creature could be a shade of the base color.
You could also have the users name their creatures, not sure how they could input the name, maybe a keyboard that takes input based on movement.

VI. Your Analysis (what you take/don't take; impact on any/all aspects of design-development)

This user testing was extremely useful and provided valuable feedback that we have used to mold and refine our current concept.

We understand that the network of data and movement of data represented by the worm is abstract. Our intentions were not to make it literal. We wanted the user to contemplate what was going on, but based on this feedback, seems like the interaction with the user is the most important, and the viewing will most likely only be interesting if the user knows where there contribution does or is currently located.

The idea that users are prompted to perform an activity was interesting. We have discussed it and would like to include something similar to this in future iterations, but not so literal. Even something as simple as a countdown of how much time is allowed for the interaction will convey much needed feedback to the user.

We are very fond of the idea that people can create their own creatures and watch them throughout the environment and will most likely implement this for our project. This is very close to the social networks that we were referencing. We are discussing whether the user should be able to choose which color (personality) they want to create. As long as people know which creature is theirs, this should retain interest in the piece beyond the initial interaction as the creatures will interact with each other.
The recommendation for using the Rock, Paper, Scissors method is a great example of simple logic that can lead to complex behaviors of the sum of parts. We will research this further, and will determine if this will fit into our creatures.

Appendix B: SuperCollider Sample Code

Foundation or Main Program, main.sc

```
//boot servers
Server.local.boot;
Server.internal.boot;
//DEFAULT VARIABLES
//list of creatures
a = List.new;
//generate a random scale
b = ScaleGen(1);
(
/*
Tangles synth
> sustain 0.1 - 1.0
< rq 0.1 - 1.0 (Actual 0.001 - 0.01)
= t_trig 0.1 - 1.0
*/
SynthDef(\tanglesSynth, {arg out=0, amp=0.5, t_trig=2, freq=100, rq=0.004, sustain=0.5,
preamp=0.2, pan = 0;
var env, signal, lag;
var rho, theta, b1, b2;
var sig;
b1 = 1.98 * 0.989999999 * cos(0.09);
b2 = 0.998057.neg;
signal = SOS.ar(K2A.ar(t_trig), 0.123, 0.0, 0.0, b1, b2);
signal = RHPF.ar(signal, freq, rq) + RHPF.ar(signal, freq*0.5, rq);
signal = Decay2.ar(signal, 0.4, 0.3, signal);
//signal = signal.wrap2 * preamp;
signal = signal * preamp;
env = EnvGen.kr(Env.perc(0.01, sustain), doneAction: 2);
```

```
DetectSilence.ar(signal, 0.01, doneAction:2);
sig = (signal * env) * amp;
sig = Pan2.ar(sig, pan, 0.5);
Out.ar(0, sig);
}).memStore;
/*
Spiky synth
> cfreq 0.1 - 1.0 (*150 = 15.5 - 675.5)
< phase 0.1 - 1.0 (*10 = 1 - 10)
= sustain 0.1 - 1.0
*/
SynthDef(\spikySynth, { arg freq = 220, amp = 0.1, sustain = 0.5, pan = 0, bits = 10, cfreq = 0.5,
phase = 0;
var sig;
sig = SinOsc.ar(freq, 0, amp) * EnvGen.kr(Env.perc(0.01, sustain), doneAction: 2);
sig = Decimator.ar(sig, SinOsc.ar(cfreq, phase, 15000, 10000), bits);
sig = RHPF.ar(sig, 600);
sig = Pan2.ar(sig, pan, 0.2);
Out.ar(0, sig);
}).memStore;
/*
Slinky synth
> cfreq1 0.1 - 1.0
< cfreq2 0.1 - 1.0
= releaseTime 0.1 - 1.0
*/
SynthDef(\slinkySynth, { arg i_out = 0, freq = 360, gate = 1, pan = 0, amp = 0.1, attack = 0,
sustain = 0.5, releaseTime = 0.5, cfreq1 = 0.25, cfreq2 = 0.3;
var out, eg, fc, osc, a, b, w, sig;
fc = LinExp.kr(LFNoise1.kr(cfreq1), -1,1,500,2000);
osc = Mix.fill(2, {SinOsc.ar(freq * [cfreq2, cfreq2], 0, amp) }) .distort* amp;
eg = EnvGen.kr(Env.perc(attack, releaseTime, 1, -4), gate, doneAction:2);
out = eg * RLPF.ar(osc, fc, 0.1);
#a, b = out;
```

```
sig = Mix.ar(a, b);
sig = Pan2.ar(sig, pan, 0.9);
Out.ar(i_out, sig);
}).memStore;
/*
Button
*/
SynthDef(\buttonClick, { arg amp = 0.1;
var sig, env, pan;
sig = Blip.ar([400, 604],  12, Pulse.ar(1.5));
env = EnvGen.kr(Env.perc(0.1, 0.5, 1, 0), 1, doneAction:2);
sig = sig * env * amp;
sig = CombN.ar(sig, 0.1, XLine.kr(0.0001, 0.01, 20), 0.2);
sig = RHPF.ar(sig, 600);
sig = Pan2.ar(sig, 0, 0.1);
Out.ar(0, sig);
}).memStore;
SynthDef(\buttonOver, { arg freq = 400, amp = 0.1;
var signal, env;
signal = Blip.ar( freq, Pulse.ar(1.5) );
env = EnvGen.kr(Env.perc(0.1, 0.5, 1, 0), 1, doneAction:2);
signal = signal.distort;
signal = signal * env * amp;
signal = Pan2.ar(signal, 0, 0.2);
Out.ar(0, signal ! 2);
}).memStore;
/* Metronome */
SynthDef(\metronome, {arg tempo=1, filterfreq=1000, rq=1.0, amp = 1, level = 0.7;
var env, signal, bass, env4, snare, hat, env3;
env3 = EnvGen.kr(Env.perc(0.002, 0.3, 1, -2), 1, doneAction:2);
hat = Klank.ar(`[ [ 6563, 9875 ],
            [ 0.6, 0.5 ],
            [ 0.002, 0.003] ], PinkNoise.ar(1));
  signal = Pan2.ar(hat*env3, 0, level);
```

```
Out.ar(0, signal);
}).memStore;
/* Limiter*/
SynthDef("limiter", {
var audio, comp;
audio = In.ar(0, 2);
Compander.ar(audio, audio,
thresh: 0.9,
slopeBelow: 1,
slopeAbove: 0.1,
clampTime: 0.01,
relaxTime: 0.01
);
}).memStore;
)
j = Synth("limiter");
j.free;
/*
buttons
0 - create
1 - tangles
2 - spiky
3 - slinky
4 - take new photo
5 - keep photo
*/
t = List.new;
//OSC RESPONDERS
//incoming osc port 57120, outgoing OSC port 12009
~netAddr = NetAddr("169.254.175.225", 12009);
c = Responders(s, ~netAddr, a, b, t); //incoming osc server, outgoing osc server, creature list,
scale, buttons
n = NetAddr("127.0.0.1", 57120);
```