Sound-Art and the Game Paradigm

Scott Simon
United State of America

Abstract

This paper focuses on a the production of a digital artwork for mobile app (IOS). The work is titled "TechnoSpaces" and it is formatted as a digital game. The paper looks at the process of building the work and some of the implications and possibilities of such processes. Key findings of the research are related to the construction of an artist's framework for mobile artworks. Various forms of creative sound manipulation and synthesis can be structured into mobile games with the right approach and vision.

Keywords

Digital art, Sound art, Music, Programming, Swift, Game, Mobile development.

Introduction

Sound-art, art, music and computer programming: these elements are the focus of the present paper. They are not separated here, one from the other. In the work of many producers (artists) all of these elements are present. The ratios may change but the basic building bricks are there. Are we artists or musicians? Writers or programmers? Many 21st Century producers are all of these things.

The present author is a composer and musician. As a musician one performs, one improvises, one composes. In many ways various other, "non-musical", cultural elements become part of the musical practice. Many musicians incorporate writing (text) and visual artistry into their activities. The manipulation of digital sound and image through programming is another important aspect of the modern musician's work. Thus one becomes perhaps, more generally, an artist.

As an artist one is constantly on the lookout for a way to develop and enhance one's métier. The present paper describes a foray into the evolution of artistic process as experienced by the artist. The focus of the paper is an artwork / game that the author is producing. The work is being done as a mobile IOS application in the Swift language.

Other researchers have looked into the connections between games and art or games and music (see Collins 2017). MoMA has acquired some games in its collection including Pac Man (Antonelli 2012). It is however a somewhat irksome topic for some (Zimmerman 2014). Yet as musicians and artists the game paradigm is of interest as it offers a wealth of frameworks and techniques that can be plundered.

The title of this paper is "Sound-art and the game paradigm". This title reveals that the focus here is upon a kind of sound-art that is, in some way, organised with or related to the game paradigm. The paper describes an experiment into the use of some game-specific components in the production of a piece of sound-art. The work that is being produced is on one level sound-art in the sense that it involves human interaction in relation to music and synthesis. It is also from another perspective an IOS game.

The paper will describe the process in its various parts. Let us list them here. (1) Make a piece of sound-art for mobile device. (2) Use various programming languages to bring this to fruition. (3) Make use of the game paradigm as a template for the work and programming.

In what follows I would like to introduce the work and briefly describe the process (section 1). Following on from that I will discuss the aesthetic outcomes in relation to programming in Swift (section 2).

Whether the completed work is ultimately art, a game, gamefied music or something else will be discussed in section 3.

TechnoSpaces

The work is an IOS application that allows the user to navigate through various screens / soundscapes. Each screen has a different visual component and these screens turn over quite quickly. The sound is comprised (in part) of a background techno track that can be manipulated through interaction with the touchscreen of the iPad or iPhone. The four corners of the screen allow a different effect or process to be activated with different ratios of mix according to finger position. Touching the middle of the screen will bring all 4 processes into play at 1/4 power (amplitude).

Added to this when a particular game task is done (for example running over an enemy object or item) the kind and amount of effect or process is changed. An example of this is as follows. If (in the first screen) the player's avatar runs over an "alien" then the entire sound file is run through a distortion, which in turn is subject to the same morphing via 4 point crossfading described above. To be precise, the crossfading continues to have an effect on the sound (filters are applied or reverb is applied) but now the entire sound is shaped via the distortion. When another alien is touched / removed the distortion ramps down. Add to this a sound effect array is triggered at each such event (created in SuperCollider). This sound effect is triggered by events such as a collision and a random sound is picked out of the array. This interplay of elements and sound triggers creates a dynamic environment.

Below I will introduce some categories that relate to the construction and make-up of the work / game. A brief discussion of (or gloss on) each category will reveal aspects of the work and the making process associated with it.

Work title: "TechnoSpaces"

Touch Screens: The general idea was to create a work that allows some kind of utilisation of the iPad or iPhone screens to manipulate music files and synthesised waveforms. The screens themselves are quite a rich resource and provide an engaging way to manipulate the sound. Vector or Crossfading synthesis is a type of modulation that changes the emphasis of a sound (e.g partials) or series of effects (mix amounts) depending upon which screen section one is located in. The middle of the screen will provide a mix of 4 different components, and any given corner one component. In between one gets various different amounts.

The audio: work out a general approach to the music and sound of the work. This could really take any form, but as I was working with Swift and Apple's core audio framework there were some constraints. I have only limited experience with Swift and therefore the first steps were a kind of experiment.

Step 1: I utilised 2 techno tracks made earlier as the raw material for the first screens. I also made use of some synthesis using Swift and Core Audio. Added to this we have interactive tasks and game actions which have their own sound effects, these were produced in SuperCollider. The SuperCollider sound effects were imported into Swift as .Wav files. From there one can use Audio Units as "nodes" through which the audio passes. Effects can be applied in this way (filters etc.), and they can be applied equally to a synthesised tone or a Wav file. Chains of these nodes can be built up and manipulated. This is done by passing an audio "node" AVAudioPlayerNode() through various AVAudioUnits. Such chains can also be built on top of audio buffers (allowing synthesis and digital signal processing).

Step 2: I modified some Swift code (Allardice 2017) to create a series of synthesised tones. The code produced in TechnoSpaces allows a chord to be built up that changes with the user's finger position (or the position of the "player").

SpriteKit / ScreenKit: SpriteKit and SceneKit are the Swift IOS frameworks for game production. For the current (first) piece SpriteKit was utilised. It is not difficult to find tutorials for using the framework, and they can be turned towards more artistic ends. There are some very useful code forms in game design that can be appropriated, and SpriteKit offers some good ways to exploit and mine these. The work TechnoSpaces involves modification and customization of code from tutorials by Todd Perkins (Perkins 2017).

Imagery: In relation to producing the imagery various approaches were utilised. To create the "sprites" png files with a transparent alpha channel were constructed in Processing. Processing is ideally situated for this kind of work. It allows the artist freedom to quickly work things through and produce the elements that are required.

In the present case I was thinking that part of the game process would include small animated "enemy" sprites. These animations are made in Swift by animating in code the png files (constructed in Processing) with a timescale between frames. One uses 4 or 5 of png files as the basis of a small animation, and this animation is then put into play in the interactive game space.

Another way of adding interactive imagery and visuals is to use Shaders. Shaders are a very interesting and exciting way of making imagery (and manipulating imagery). Shaders (written in GLSL) are a type of program that makes use of the OpenGL framework. These shaders can easily be embedded in the Swift IOS system. The advantage of using these is that the visual then runs on the GPU of the device. This makes creating interesting and sophisticated visuals less CPU intensive, also allowing the programmer to keep the frame-rate of the application high. Shaders can be passed into an SKScene in Swift with the SKshader class which takes one parameter (the name of the shader itself: SKShader (fileNamed: "shader").

Aesthetics and Swift

The aesthetic outcomes of the "TechnoSpaces" application are multiple. The app allows one to navigate through a series of screens in which music files are manipulated and visual tasks accomplished. The difficulty factor is not extreme: it is conceived as an enjoyable way to interact with sound and imagery. Each screen involves a background openGL structure that has some "movement". On this moving screen are embedded sprites and the "player". The touchscreen controls the "player", its avatar following the user's finger. The music and sound changes and is subject to the same gestures controlling the player. Other interactive sounds also appear and disappear within the sonic landscape.

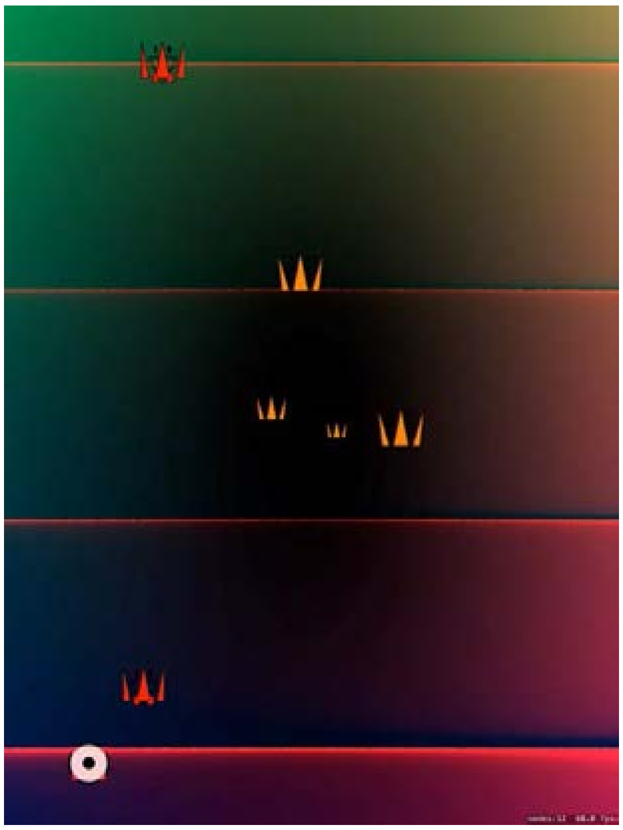

Figure 1. Screen from TechnoSpaces

The different classes governing each screen are written in Swift. The classes have some global components that run above the changing screens - for example the background techno soundtrack remains through the first series of screens. The manner in which it is manipulated / modulated changes with each screen.

Discussion

The working process described here is still in a nascent form. It is a step towards more complex interactions and sound manipulation. That said, even as it structured now it could be utilised as a performance tool that has some quite expressive capabilities. The touchscreen naturally lends itself to quick and fluid motions and the tracking is excellent. Using the work as a performance tool requires that everything be created with an eye (or ear) towards detailed musical and tonal quality. I have found that the parts are in place to allow such a focus on detail. Some extra yards are required in relation to the programming to ensure such results.

An aspect that could be emphasised is the rhythmic component of the interactions with the music. This would involve a task which can only be completed "on the beat" as it were. Thus the gesture would have to synchronise with the rhythms of the audio. This can be implemented quite easily by inserting a "tap" onto the output of Core Audio class. Such a tap could then be utilised as a test for collision or other captured gestures.

In relation to the status of the work and which genre it fits into, a couple of points can be made. At the moment the overall structure of the work make it very "game-like", and it is approached in this spirit often. However the author feels that it could easily become a performance tool for Electronic music, and it fits nicely into this category. As it stands it can be used in such a manner - adding bluetooth connectivity with a good pair of speakers fill out such a system.

Conclusion

Working in the field of mobile applications is a rewarding and exciting direction for artists and musicians to go in. However it is not always easy to know where to start. The present research offers a completed work for discussion and analysis. The framework is built from various different technologies that can be understood and used by all. It is often difficult to know how to combine the different technologies into a good and meaningful practice that will reward use. The present research presents an option, or series of options, that can solve some of these dilemmas.

The piece is currently in beta testing on Apple's "TestFlight". Interested parties can contact the author to be added to the tester's program.

The source code will be available for study / reference at the conference, the author will construct a gist for the purpose.

Acknowledgements

The author wishes to thank the Swift community for providing some of the answers to programming questions and CCS in Sydney for taking part in the Beta testing.

Thanks also go to the Interactive Media students at UTS for their great ideas and slick programming solutions.

References

Allardice, S. (2017). *Code Clinic: Swift.* Retrieved from https://www.lynda.com/Swift-tutorials/Code-ClinicSwift/362874-2.html?

Antonelli, P. (2012). *Video games: 14 in the collection for starters*. Retrieved from https://www.moma.org/explore/inside_out/2012/11/29/video-games-14-in-the-collection-for-starters/

Collins, K. (2017). *From Pac-Man to Pop Music: Interactive Audio in Games and New Media*, London: Routledge.

Perkins, T. (2017). *IOS game development with Swift 3 and SpriteKit*. Retrieved from https://www.lynda.com/Swift-tutorials/iOS-Game-Development-Swift-3-SpriteKit/512723-2.html

Zimmerman, E. (2014). *Games stay away from art, please.* Retrieved from https://www.polygon.com/2014/9/10/6101639/games-art