iOS Audio Programming: Clum.sy Technical Report

Department of Electronic Engineering, University of York

Y3863041

January 23, 2023

1 Introduction

This report documents the design and development of *clum.sy*, an iOS app for iPads that uses physics simulations and randomized chord progressions to create an interactive musical experience. The design process started with researching and taking inspiration from the functionality and UI of compelling apps and VSTs. This was followed by a development phase that continued until the app's core functionality was integrated, at which point extensive testing began to ensure a smooth implementation of any additional features. This report provides an overview of the app's features, a detailed description of the design and development process, information on the testing and evaluation process, as well as the future scope and marketing of the project.

2 Functionality

clum.sy uses simulated gravity and randomised chord progressions to provide the user with a unique and evolving musical experience each time the app is used. The app's screen is split into two halves, with a menu bar at the top which can also access a settings menu. The left half of the screen controls the bass notes and the harmony, and the right half controls the timing and velocity of the notes.

The left half of the screen hosts a ball that will float about and randomly collide with a "chord bouncer" object that represents a scale degree of the C major scale. When the ball strikes a bouncer, a MIDI note corresponding to that scale degree will be played and a variable known as "activeChord" is set to an integer value representing that scale degree. The value of this variable dictates what chord will be played by the right half of the screen.

The right side of the screen is comprised of four piano keys that represent the 1st, 3rd, 5th, and 7th degrees of a chord. Up to four balls can then be spawned anywhere on this right half of the screen that upon colliding with the piano keys tells the sampler to play MIDI notes. The velocity with which the balls strike the keys also correlates to the MIDI velocity with which the notes are played. The previously mentioned, "activeChord" variable changes what MIDI notes are played by the piano keys. If "activeChord" equals 0 (ie 1), then the chord is I, which in C major is C, E, G, B. If "activeChord" equals 4 (ie 5), then the chord is V, so G, B, D, F, etc.

The menu bar at the top provides quick-access to useful settings like changing the sound effects, play/pausing the app, reading the information page, and then accessing the settings menu. The settings menu provides more in depth access to a wider array of settings and parameters, such as choosing which chord bouncers are present on the screen, in depth access to effects parameters and the ability to change sounds, and access to physics settings.

3 Features

clum.sy uses a variety of audio effects and physics settings to create a uniquely relaxing sonic atmosphere. Given the prerequisite knowledge of chord degrees, the app is targeted towards professional and hobbyist musicians. It is intended to serve as a musical "fidget" tool that provides the user with a mental break or help to pass the time, though there is a distinct potential for use in music production which will be explored in the Future development section. The app's features and customisation options were purposefully limited to maintain a minimalist look and intuitive user experience. These features and options included in the app are:

- Settings menu
- Sound selection & Musical effects
- Physics settings
- Sleek design & Intuitive UI

Settings Menu: *clum.sy* includes a settings menu housed within the GameScene instead of a separate Scene so that the various physics and musical effects settings can be adjusted whilst the game is still running.

Sound selection & Musical effects: Six different musical tones are available to choose from within the settings menu, from a simple grand piano, to more percussive, ASMR-based plucks. Filter, delay, and reverb effects are included in the app. The parameters of the filter and reverb can be dynamically adjusted by moving a finger around the entire screen whilst the effect is currently selected. This also includes a small animation to visually alert the user that the effect is being manipulated. Control of the delay effect on the other hand is contained within the settings menu due to stability issues using it the other way. The ability to turn on and off these three effects is also stored within the settings menu.

Physics Settings: The settings menu also includes various physics settings that affect the dynamics of the balls, adding an additional layer of interactivity and engagement for the user. There is a "speed" parameter that affects the speed of the "bassBall" on the left side of the screen. This essentially modifies how often the active chord will change depending on how many chord bouncers are present. There is also a "gravity" parameter that alters the gravity of the piano balls on the right side of the screen. As a result, the frequency and impact velocity of the balls striking the piano keys is also affected. A reset button is also included to clean up the scene if necessecary.

Sleek Design & Intuitive UI: *clum.sy* has been designed to minimise as much settings information as possible. This meant only displaying essential options/buttons on the main screen and hiding non-essential parameters within the settings menu or using the screen (eg filter and reverb effects). This has been done to encourage the user to explore the app and experiment for themselves, cultivating a more interactive and personable user experience.

4 Design & development

Before deciding on the direction for the app, several VSTs and musical apps were explored and their most captivating and engaging features were written down. One particular VST, Ableton's Bouncy Notes[1], provided most of the inspiration behind the core functionality of *clum.sy*.

As the development of *clum.sy* progressed, the design process behind it evolved organically. Multiple prototypes were trialled, experimenting with different methods for sound generation and physics simulations. Ultimately, the final design incorporated AudioKit's "MIDISampler" class for audio and Sprite Kit's SKPhysicsBody class to simuate physics. The app's layout also evolved dynamically. Simple objects such as dividers, labels, and shapes were positioned and sized according to the 1024x768 resolution of the iPad. For more complex designs like the settings menu, ideas were first sketched out digitally to experiment with and optimise placements.

4.1 Model-View-Controller

The majority of the app development took place in four files: "GameScene.swift," "GameViewController.swift," "Sampler.swift," and "Main.storyboard." *clum.sy* follows the traditional Model-View-Contoller (MVC) design pattern and separates these files into the three distinct components.

The model consists of "GameScene.swift" and "Sampler.swift." "GameScene.swift" contains the core functionality of the app, including the simulation of physics using Sprite Kit, procedures for trigger events such as collisions and touches on the screen, and functions for altering the physics settings of the scene. The "Sampler.swift" file uses AudioKit's "AudioEngine", "MIDISampler", "LowPassFilter", "Delay", and "Reverb" classes to produce the audio for the app. It plays the MIDI notes of the sound file that is currently selected using 20 different functions, each of which plays a unique note at a specified MIDI velocity. There are also several functions used to alter the parameters of the three effects.

The app's view is split between "Main.storyboard" and "GameScene.swift". Initially, it was solely set up in "GameScene.swift" using SwiftUI and Sprite Kit in an effort to prevent crashes that occured when used with "Main.storyboard". This issue was subsequently resolved and the storyboard then employed to hold elements linked to the clike UISliders. However, the fix was implemented towards in the end of development, so there remains a significant amount of code that could benefit from refactoring. This will be further evaluated in the next stage of development.

Finally, the controller is again split between "GameScene.swift" and "GameViewController.swift". "GameScene.swift" contains overwritten functions such as "touchesBegan" and "touchesMoved" to handle touches anywhere in the scene. This can include touching nodes such as the settings label or the empty space used to spawn a ball. "GameViewController.swift" is more exact and only handles functions that are executed when UI buttons and sliders housed in the settings menu are altered, such as changing the scene's gravity when its slider is touched.

5 Future development

The current version of *clum.sy* establishes a strong foundation for a distinctive and compelling musical application. If development were not limited to this current version, there are several areas that could be improved and expanded upon in the future.

The current codebase could be refactored to improve readability, maintainability and scalability of the app. This would primarily focus on restructuring "GameScene.swift", removing redundancies and implementing design patterns to make the code more modular. A specific example of this would be replacing two SKNode classes used to implement the chord buttons with UIButton classes. This would convert a significant amount of code used to set up and provide functionality for the chord buttons to visual objects in "Main.storyboard". Less critical improvements would include using more custom classes to dynamically create objects and grouping the effects and their settings together.

Several new features could be added to the app to enhance its functionality and improve the user experience. An immediate upgrade would be adding the option to customise the chords and locations of the bouncers, as well as the ability to upload custom sounds or download samples from an online community. Moreover, integrating social media and the ability to share creations with friends would be a valuable marketing tool.

Finally, incorporating features such as quantization and expanded control over the rhythm and tone of the app could enhance its functionality and make it a valuable tool for music production and digital audio workstations (DAWs). In that scenario, porting the app as a virtual studio technology (VST) would be a beneficial addition.

6 Marketing strategy

Due to the purposefully limited nature of *clum.sy*, it is intended to be available for free on the App Store, with the potential to host ads beneath the settings menu in the future. This could generate revenue without interrupting the user's experience. An additional method of monetising the app could be the implementation of in-app purchases. This could involve offering supplementary content or new features that can be unlocked through purchase within the app.

In order to effectively target the intended audience, a multifaceted approach to marketing should be employed. Utilising social media platforms, such as Facebook and Instagram, as well as influencer marketing and search engine optimization techniques, can aid in reaching the target demographic of professional and hobbyist musicians.

References

[1] Ableton, "Learn live 11: Bouncy notes," Jun 2021. [Online]. Available: https://www.youtube.com/watch?v=C2hQ-WbKBhU