Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

- Java Code Conventions

Summary of Requirements

List all requirements as bullet points in brief.

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- Com S 227/228 - introduction to Java programming
- EE 224 (Signals & Systems I) - introduces the Fourier transform and use of MATLAB for digital signal processing
- EE 285 - introduction to good coding practices using C
- EE 321 - further explored signal process concepts and working with Simulink

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- Android development
- Generation of envelope functions
- Audio pitch-shifting

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

- Drs. Randall Geiger and Degang Chen served an advisory role and offered technical advice.

1.2 Problem and Project Statement

While there exist fully-featured samplers on iOS, samplers on Android have a limited feature set. Many of the samplers on the market do not feature an envelope function, for example. Such a function is used heavily in the world of samplers.

As well, existing sampler applications for smart devices fail to offer users an intuitive view of the effects of their actions. Users can work with the applications, but need to rely on external documentation and knowledge to understand how to use them.

The existing sampler apps for Android This project is a fully-featured Android sampler app geared around offering a clear picture of what is happening and an accessible UI.

1.3 Operational Environment

The end product will be an application for Android devices. The operational environment is a tablet, phone, or other smart device running the Android OS. One of the constraints that comes with this is the wide variation in processing power between different devices.

1.4 Requirements

Functional requirements

- The application must be able to run on a fairly modern Android device
- The application must have an easy-to-use, intuitive UI
- The product must sound good to musicians
- The application must have the following features:
    - Pitch-shifting
    - Speed-shifting
    - Envelopes
    - Adjustable band-pass filter
    - Ability to record and save application output
    - Haptic feedback
    - Ability to save and load presets

Economic requirements

- The application must conform to the requirements of the Google Play store
- The application needs to have an appealing sound to musicians, both hobbyist and professional

1.5 Intended Users and Uses

The product has two major user groups: hobbyist and performing musicians. Hobbyist musicians will use the app as a more robust version of existing sampler apps that better substitutes for a physical sampler. Performing musicians will use the app to create and perform music on a more portable device.

1.6 Assumptions and Limitations

Assumptions:

- The application will run on a fairly modern (released within last five years) Android device
- The maximum number of simultaneous inputs to the app is less than ten

Limitations

- The application will need to be lightweight enough to run on phones
- The application will need to output audio in a format compatible with how Android interfaces with speakers

1.7 Expected End Product and Deliverables

End product: sampler application for Android
Delivery date: week of April 27, 2019
The end product will be a sampler application for Android with the features listed in section 1.7. It will also include refinements based on feedback from alpha and beta testing.

Deliverable: equalizer component
Delivery date: week of February 23, 2020
This component will be the code and UI to apply a user-drawn equalizer curve to a piece of audio.

Deliverable: speed-shifting component

Delivery date: week of January 20, 2020
This component will be the code to change the speed of a piece of audio.

Deliverable: pitch-shifting component
Delivery date: week of February 10, 2020
This component will be the ability to change the pitch of a piece of audio while maintaining its tempo.

Deliverable: envelope component
Delivery date: week of February 3, 2020
This component will be the code and UI to apply a user-defined ADSR or ADHSR envelope to a piece of audio.

Deliverable: UI
Delivery date: week of March 9, 2020
This component will be the user interface that ties together all the app's functions.

Deliverable: feedback from beta testing
Delivery date: week of March 23, 2020
Beta testing will take place with a group of musicians and their feedback will be recorded.

2. Specifications and Analysis

2.1 Proposed Design

Include any/all possible methods of approach to solving the problem:

– Discuss what you have done so far – what have you tried/implemented/tested, etc?

– We want to know what you have done

• Approach methods should be inclusive of **functional and non-functional requirements** of the project, which can be repeated or just referred to in this section

If your project is relevant to any **standards** (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here

The proposed design is an Android sampler app with a simplified UI that increases the amount of user control. The application will be split into several subcomponents which are controlled either from the main screen of the app or from their own screen, depending on their size. The application will also have a dedicated "play" mode that hides and disables editing features to save screen space.

2.2 Design Analysis

– Discuss what you did so far

– Did it work? Why or why not?

– What are your observations, thoughts, and ideas to modify or continue?

– If you have key results they may be included here or in the separate “Results” section

-Highlight the **strengths, weakness**, and your observations made on the proposed solution.

So far, nothing has been created or tested. However, there are still several changes that have been made:

- Per feedback from Dr. Geiger, the app’s scope was changed to only be a sampler. Initially, the app would’ve been controllable via guitar, but the group worried the scope was too large, as did Dr. Geiger. The approach of removing the guitar from the equation allows a functional sampler to be created that another group could later add guitar control to.
- Initially, the application would not have a separate mode just for playback.

2.3 Development Process

Discuss what development process you are following with a rationale for it – Waterfall, TDD, Agile. Note that this is not necessarily only for software projects. Development processes are applicable for all design projects.

The project will follow a Waterfall process. The main reason for this decision is the small group size. Because the group only has three members and only one member has Java experience, the group will often be working on the same thing together. The linear workflow of the Waterfall process fits well with this.

2.4 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

3. Statement of Work

3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to “compete” with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

The Akai iMPC sampler app is no longer available for Android as of October 2019. [1]

3.2 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

Android Studio

Strengths:

- Intuitive UI that allows all members to contribute
- Simplifies UI creation
- Commonly used for Android Development
- Features a debugger

Weaknesses:

- Requires the members to learn a new IDE

Android Studio was chosen over other Java IDEs because of its overall simplicity. The way it simplifies creating a UI was a large plus given the team’s lack of experience in Android

development. As well, it provides similar features to many other IDEs. Even though the team is more experienced with Eclipse, Android Studio provided a large number of benefits.

Monkey

Strengths:

- Allows for stress-testing the app via pseudo-random inputs
- Pseudo-random nature allows for tests to be repeated

Weaknesses:

- Does not target specific cases

3.3 Task Decomposition

Most of the tasks in this project require that we have a working way to load audio from the Android file system, then play it from our app's memory. Loading and playing audio will serve as the backbone of the app. Once that works, the audio processing tasks can be implemented individually. These tasks are the pitch-shifting, speed-shifting, filtering, and envelope functions. Then, they need to be combined and tested to ensure that they function alongside each other. Finally, they and the playback features need to be implemented together into a UI.

3.4 Possible Risks And Risk Management

One risk that our project faces is that the implementations of our equalizer and pitch-shifter could be more complex than initially envisioned. To handle this, the communications/signal-processing-focused students can consult with professors in the area. The equalizer is less likely to be an area of risk than the pitch-shifter as the students have experience working with equalizers.

Another risk is our team's limited Java programming experience. To mitigate this, the team is using Android Studio for the app development because it is a fairly simple IDE. Also, the team will be making heavy use of pseudocode and paired programming. This also serves the purpose of allowing the most experienced coder of the group to receive help in understanding signal-processing functions.

3.5 Project Proposed Milestones and Evaluation Criteria

The first key milestone is having a functional equalizer. In order to test the equalizer functionality, we plan to compare the drawn function shape to the actual shape. We also plan to test a series of random inputs using Monkey.

The second key milestone is having a working sample speed shifter. This can be tested using samples of various lengths and comparing the output of our app to a working MATLAB equivalent.

The third milestone is implementing frequency-shifting. This will be testing again using a set of samples of various lengths, initial frequencies, and amplitudes. This will let us cover a wide variety of types of inputs.

3.6 Project Tracking Procedures

Our group will use Google Drive and Git to track progress. Slack will be used to coordinate weekly meetings.

3.7 Expected Results and Validation

The desired outcome is that we would be able to create a sampler app that lets users understand and feel involved in the process of performing with a sampler. Our plan to ensure our solutions work is to perform beta and final testing with performing and hobbyist musicians.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

• A realistic, well-planned schedule is an essential component of every well-planned project

• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

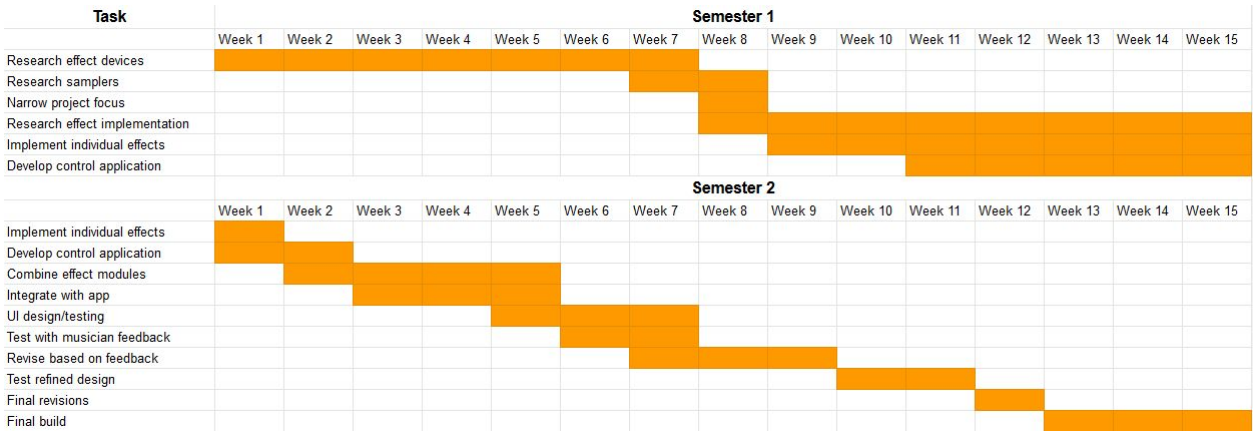
– Start with a Gantt chart showing the tasks (that you developed in 3.3) and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.

– Annotate the Gantt chart with when each project deliverable will be delivered

• Completely compatible with an Agile development cycle if that's your thing

How would you plan for the project to be completed in two semesters? Represent with appropriate charts and tables or other means.

Make sure to include at least a couple paragraphs discussing the timeline and why it is being proposed. Include details that distinguish between design details for present project version and later stages of project.



To-do: replace Gantt chart with version based on final design plan

4.2 Feasibility Assessment

Realistic projection of what the project will be. State foreseen challenges of the project.

Challenges that will be faced throughout the project process include implementing envelopes, UI design, and working with Java.

4.3 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be based on the

projected effort required to perform the task correctly and not just “X” hours per week for the number of weeks that the task is active

4.4 Other Resource Requirements

- Computers with the Android Studio IDE
    - The group members will be using their personal computers

No other resource requirements are predicted.

4.5 Financial Requirements

As Android Studio is free software, no financial resources will be needed to conduct the project.

5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case 5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you’ve determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

5.2 Hardware and software

Monkey is a piece of software that allows a pseudo-random set of inputs to be sent to an app for testing purposes.

5.3 Functional Testing

- The application will be compared to pre existing software for PC that performs similar effects
- Expected output and obtained output will be compared for a variety of length, amplitude, and frequency content values
- The app will be tested with musicians to obtain feedback and make any necessary changes

5.4 Non-Functional Testing

- The app will be tested with musicians to obtain feedback and make any necessary changes
- Monkey will be used to test the app against a random series of inputs

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

5.6 Results

– List and explain any and all results obtained so far during the testing phase

- – Include failures and successes
- – Explain what you learned and how you are planning to change it as you progress with your project
- – If you are including figures, please include captions and cite it in the text

• This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

No testing has been performed at this stage in the design process.

6. Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

At this stage in the process, while no progress has been made on the creation side of things, the group has a strong design framework. Much of the app is already planned and designed, so the future leaves only implementation and testing. Creating a fully-functional Android sampler app with an intuitive UI addresses the two largest shortcomings of the currently available apps.

6.2 References

This will likely be different than in project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

[1] “iMPC for Android,” *AKAI Professional*. [Online]. Available: https://www.akaipro.com/impc-for-android. [Accessed: Nov. 18, 2019].

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.