Music Tool
Design Document

Jackson Dean - Isaac Gibson - Xiaodi Jiang - Carter Morgan

CS 4000 - Senior Capstone Design

March 14, 2022

Summary

Virtual synthesizers and sequencing software have lowered the barrier of entry for people to make interesting, high-quality music. Innovative interfaces give inexperienced people a chance to experience the joy of interactive music creation by stripping away some of the “hard” parts of making music, as well as giving inspiration to advanced musicians by allowing them to interact with the sound in a new way. Our project will be a new interactive audio-visual art piece inspired by the 2000’s “Animusic” music video series.

The user will be able to create new beats and tracks using a physics-based “marble machine” interface where the sound is emitted by interactions between the virtual instruments including percussion, strings, electronic synthesizers, and other abstract objects. This interface allows the user to simultaneously create a piece of music as well as an interesting animation. It will allow them to intuitively create music while also encouraging them to be creative in ways they wouldn’t expect.

Creation of the audio visual experience will require expertise in web graphics, physics, audio manipulation, and UI/UX design. Our stretch goals would also include aspects of database design and live multi-user google doc style networking. The combined effort from the team over the next few months will result in an application that enables anyone to be a musician.

Background/Technical Requirements

Overview

The system we are proposing would have an interface where it is obvious exactly how the user's actions translate to sound since they are manipulating "physical" objects. This system should be intuitive for users to understand and use.

The goal of the system is not to be as capable as a full-fledged audio workstation, but should allow more intricate pieces to be made unlike with many intentionally simple applications that limit the variety of music possible.

The project will be a stunning display of projectiles and instruments giving users the creative freedom to create music in a visually entertaining manner with ease of making the auditory experience they want. The user interface will be designed to complement the main artwork. Users will also be impressed with what has been made and have the option to use social media to show friends the musical marble masterpiece.

We plan to use complex web graphics and physics simulations that will be synchronized with the audio manipulation aspects of the music. Throughout the design we will also be demonstrating our skill with creating an interesting and well built website and use of a database to store creations, which can help users manage and share their artworks conveniently. What's more, our web application will provide a good music-related social platform that allows our users to group up and cooperate online simultaneously.

Technologies

Azure

Azure is a public cloud computing platform. We plan to use Azure to hold our remote server and database on the web. We plan to use Visual Studio to publish our web application directly to Azure server.

React

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. We plan to compose complex UIs from small and isolated pieces of code called “components” with React in our web application. We will learn React from official tutorials and create most of our front-end code to implement our application features.

MatterJS

Matter.js is a 2D rigid body physics engine written in JavaScript. Matter.js builds the physical world using rigid bodies and simulates physical properties such as gravity and friction to represent the marble machine. We will use Matter.js to support our collisions and physical properties in 2D canvas for our application.

PixiJS

Pixi is an inclusive technology and all content can be made to be screen reader accessible with ease. We decided to use Pixi.js becauseMatter.js’s built-in renderer is mostly just capable of rendering objects as they appear in the simulation, which is useful for debugging, but not for a finished product. Pixi.js offers a number of benefits, so we plan to use Pixi to help our application render faster and implement more complex visual effects, including animations.

ToneJS

Tone.js is a Web Audio framework for creating interactive music in the browser. All of the sound, as well as event scheduling, etc, is handled by the Tone.js library. It is a very extensive synthesizer library that in many ways functions just like an analog synthesizer. We plan to create sound sources, pass the signal through a chain of filters and effects by using Tone.js. Then connect it to the computer’s output. Everything about this chain is customizable, so in the future a lot of work will go into exposing these options to the user in a graphical interface.

React DND

React DnD is a set of React utilities to help users build complex drag and drop interfaces while keeping react components decoupled. We plan to use React-DnD to implement image drag and drop features in React. By making use of useDrag() and useDrop() functions, we will achieve image dragging, placing and layout positioning faster and easier.

Software/Hardware

Github

GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to code. We use Github to track our code that implements different application features separately. Also, we use the github project board to help us track TODO issues, which allows us to move forward with each task in an organized manner.

Postman

Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated. We will use postman to help us test and debug our database.

Documentation

Our documentation will help us keep track of the configuration of ASP NET.Core and Azure services. Also, it will become a part of our application introduction to help users know better about our project.

Requirements Analysis

System Architecture

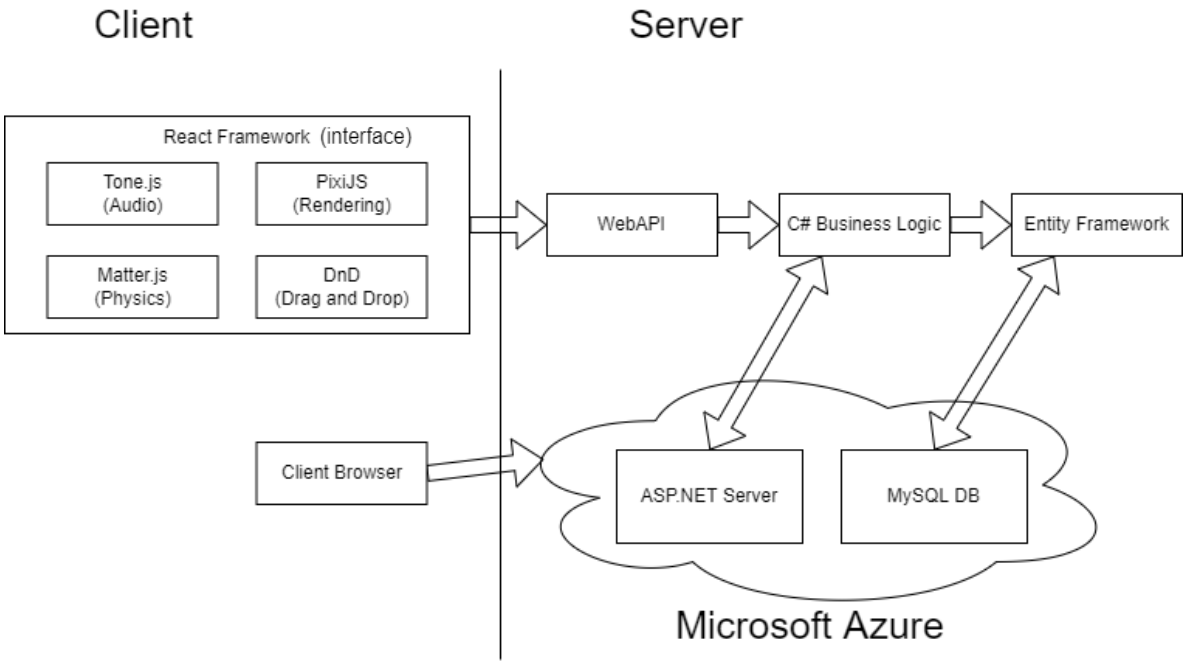

Figure 0: System architecture sketch

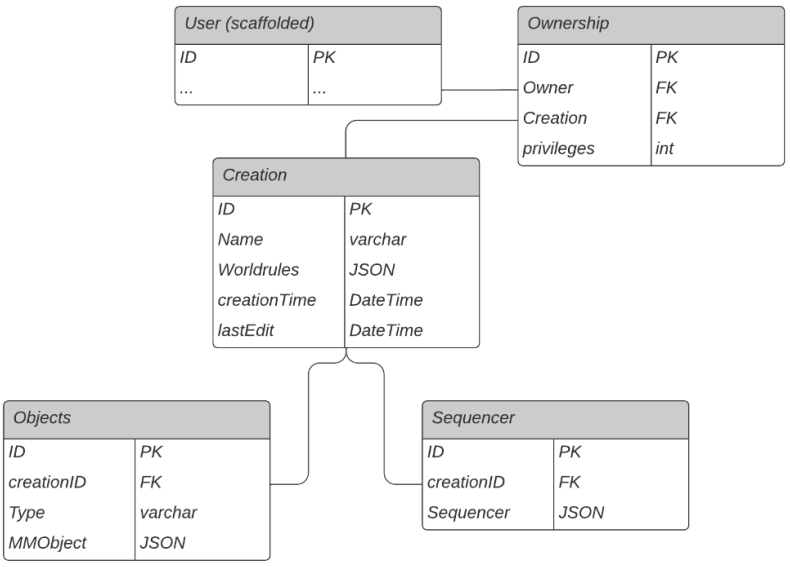

Figure 1: Database's JSON String schema architecture sketch

Web Application Overview

As Figure 0 shows, our web application is made up of a front-end client side and a back-end server side. The front-end client side is built on React Framework, we implement and combine several music tool features using react components. The back-end server side is held on Microsoft Azure, where we publish the website and store our database.

Audio and 2D Cannon Scene

The front-end application will provide users with the freedom to create their own melodies, which can be looped by selecting different nodes on the sequencer. Each cannon in the 2D scene is bound to a node track in the sequencer, when it is the selected node's turn to make sound, the corresponding cannon shoots a ball. In this case, we complete the combination of picture and audio.

User Data Storage

In order to keep our database's tables simple, we plan to use JSON strings to store user information and user's 2D scene project information. Figure 1 shows the idea on the schema for saving JSON. We let the JSON string help us track all the user information along with their project information. In this case, when users want to load their previous projects, we find their corresponding JSON strings from the database table, then read the JSON and get the properties of the cannons and instruments, we reload the audio nodes and redraw the 2D scene based on these properties after that. Also, it is worth mentioning that the idea of ID and creationID is basically objects and sequences belonging to a creation and creations belonging to one or more users.

The JSON strings stored for "Objects" in figure 1 represent physical objects in the scene including cannons and instruments. The stored JSON contains a simplified version of the objects including what type of object it is, the position data, what sound(s) it makes and options for controlling physical interactions. For "Sequencer" stored JSON it contains the information for how many tracks a stored sequencer has as well as the stored musical score created with the sequencer.

Personnel

- **Web Based Programming and Design**
  - Jackson Dean
  - Isaac Gibson
  - Carter Morgan
- **Database Design and Management**
  - Carter Morgan
  - Jackson Dean
  - Isaac Gibson
  - Xiaodi Jiang
- **Computer Graphics and Physics Motion**
  - Jackson Dean
  - Isaac Gibson
  - Carter Morgan
  - Xiaodi Jiang
- **Audio Handling**
  - Xiaodi Jiang
- **Visual Design**
  - Jackson Dean
  - Xiaodi Jiang
  - Isaac Gibson

System Features

Rank 1: Bare Essentials

- Rhythm Sequencing
- Cannon Manipulation
  - Selection and moving
- Note Block Manipulation
  - Selection and moving
- Component Pallet
- Object Interactions
- Hosted web page
- User Accounts
- Project Storage

Rank 2: Planned Features

- Project Exporting / Sharing
- Animations coupled with sound
- Advanced timing / trajectory control
- Customisable sounds
- Tutorials / Examples
- Multiplayer collaboration
  - store objects
  - no communications during ‘play music mode’
  - communicate changes
  - only one person can select a object
- Social Media Integration

Rank 3: Bells and Whistles

- Advanced project management
  - Save “blocks,” templates, etc
  - Ctrl-Z Undo button
- Upload samples
- Community project explorer
- Custom shapes / animations
- Advanced physics control
- Many prebuilt instruments
- Project Layers
  - Change layering
- Sequencing changes to audio properties
- External Controller Input (MIDI)
- Game aspects
  - Challenges
  - Leaderboards

Timeline

Week	Jackson Dean	Isaac Gibson	Carter Morgan	Xiaodi Jiang
1	Reset Azure	Clean up branches	Make team report website	Clean up branches
2	Configure database	Fix scene bugs	Combine components	Connect user account to database
3	Fix scene bugs	Fix scene bugs	Combine components along with Pixi	Connect user account to database
4	Bugs	Bugs	Bugs	Bugs
5	Combine components and Polish	Combine components and Polish	Combine components and Polish	Cooperative creation support demo
6	Associate projects with user accounts	Export and upload projects as JSON files	Export and upload projects as JSON files	Associate projects with user accounts
7	Save project to database	Save project to database	Save project to database	Save project to database
8	Bugs	Bugs	Bugs	Cooperative creation support demo
9	Combine components and polish	Combine components and polish	Combine components and polish	Cooperative creation support
10	Built-in tutorial	Built-in tutorial	Built-in tutorial	Built-in tutorial
11	Project explorer	Beautify UI appearance	Project explorer	Project explorer
12	Project explorer	Beautify UI appearance	Project explorer	Project explorer
13	Bugs	Bugs	Bugs	Bugs
14	Polish	Polish	Polish	Polish

15	Easter Eggs	Easter Eggs	Easter Eggs	Easter Eggs

Appendix A: UI Sketches

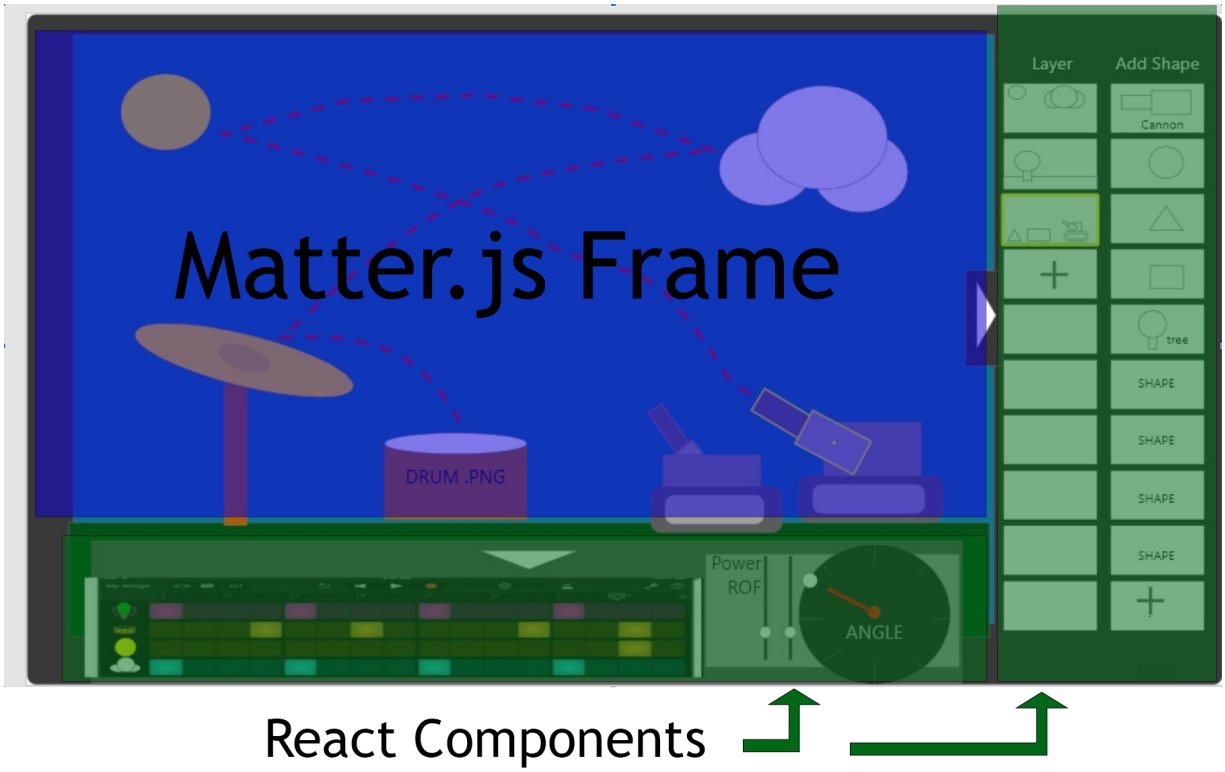

Figure 2: 2D cannon scene sketch for music tool

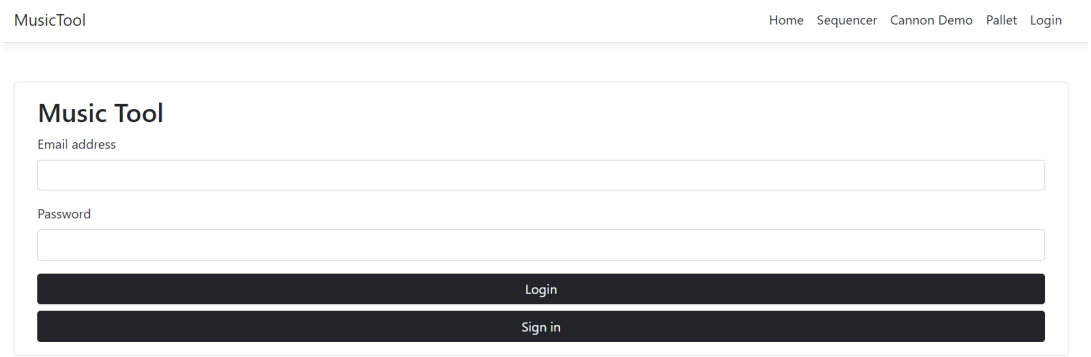

Figure 3: Login and sign in page

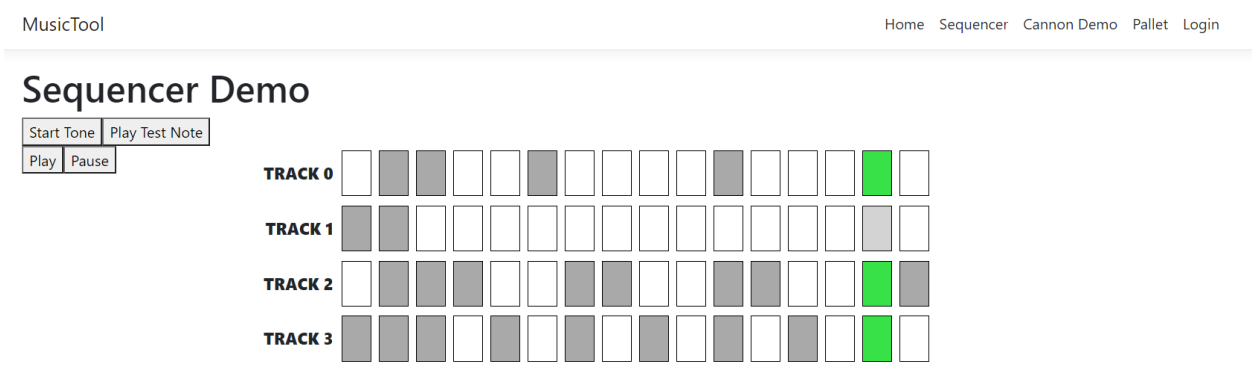

Figure 4: Audio demo

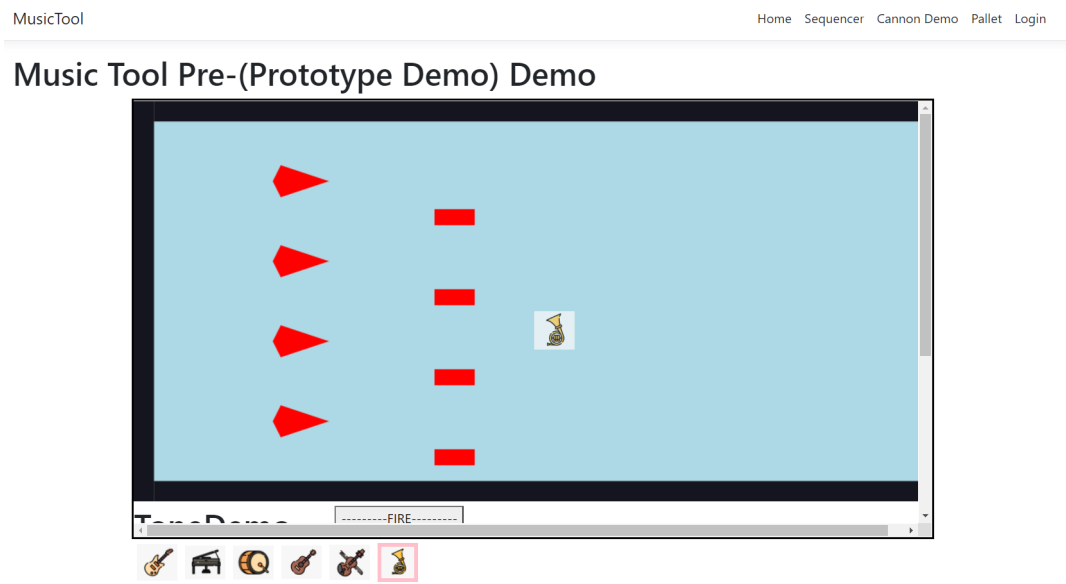

Figure 5: Musical instrument drag and drop pallet demo

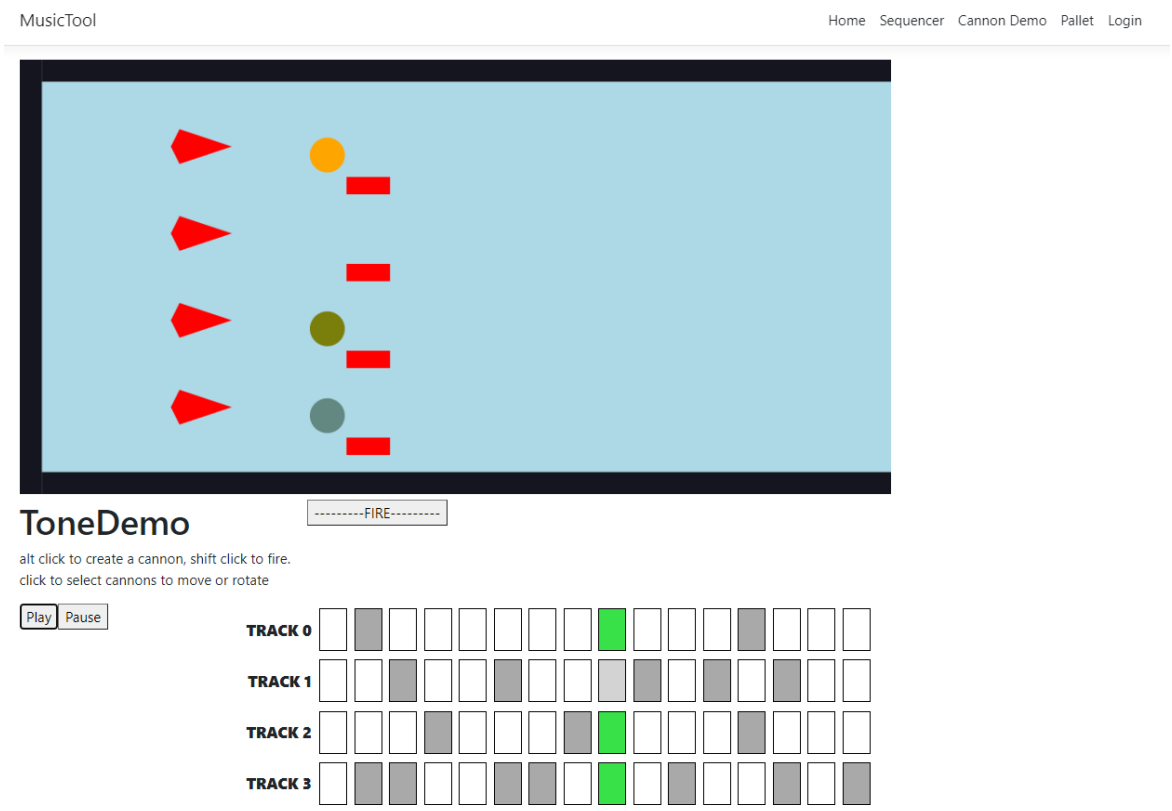

Figure 6: Combination of audio demo and cannon demo

Appendix B: Use Cases

Use Case Number	1
Title	Login/Sign in user account
Description	User want to login to our website
Steps	1. Users navigate to our website https://music-tool.azurewebsites.net/ 2. For new users, enter the email address and password then click sign in. For old users, enter the correct email address and password then click login.
Related Figure	Figure 3

Use Case Number	2
Title	User modify tone and nodes on sequencer
Description	User want to compose a melody
Steps	1. Users select and click several nodes on different tracks 2. Users hit “play” button to let the melody play in a loop
Related Figure	Figure 4

Use Case Number	3
Title	User drag and drop musical instrument
Description	User want to drag an instrument onto the 2D canvas and let it make sounds
Steps	1. Users click on the image and hold their mouse 2. Users move their mouse and drop the image onto the canvas at anywhere they want
Related Figure	Figure 5

Use Case Number	4
Title	User play with cannon to shoot balls and make sounds
Description	User want to set cannons and let them shoot balls and make sounds when a collision happen
Steps	1. Users click on cannon and hold on mouse to change the angle of cannon 2. Users select nodes on sequencer 3. Users hit play, then each time when a ball is shot by a cannon, the corresponding node in sequencer makes sound. 4. The melody made by sequence plays in a loop
Related Figure	Figure 6