Music Tool
Design Document

Jackson Dean - Isaac Gibson - Xiaodi Jiang - Carter Morgan

CS 4000 - Senior Capstone Design

November 4, 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.

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.

Fetch

The Fetch API provides an interface for fetching resources (including across the network). It provides a more powerful and flexible feature set than XMLHttpRequest. Fetch provides a generic definition of Request and Response objects, which allows them to be used wherever they are needed in the future, whether it's for service workers, Cache API, and other similar things that handle or modify requests and responses, or any kind of use case that might require you to generate the responses programmatically. It also defines related concepts such as CORS and the HTTP Origin header semantics, supplanting the separate definitions elsewhere.

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.
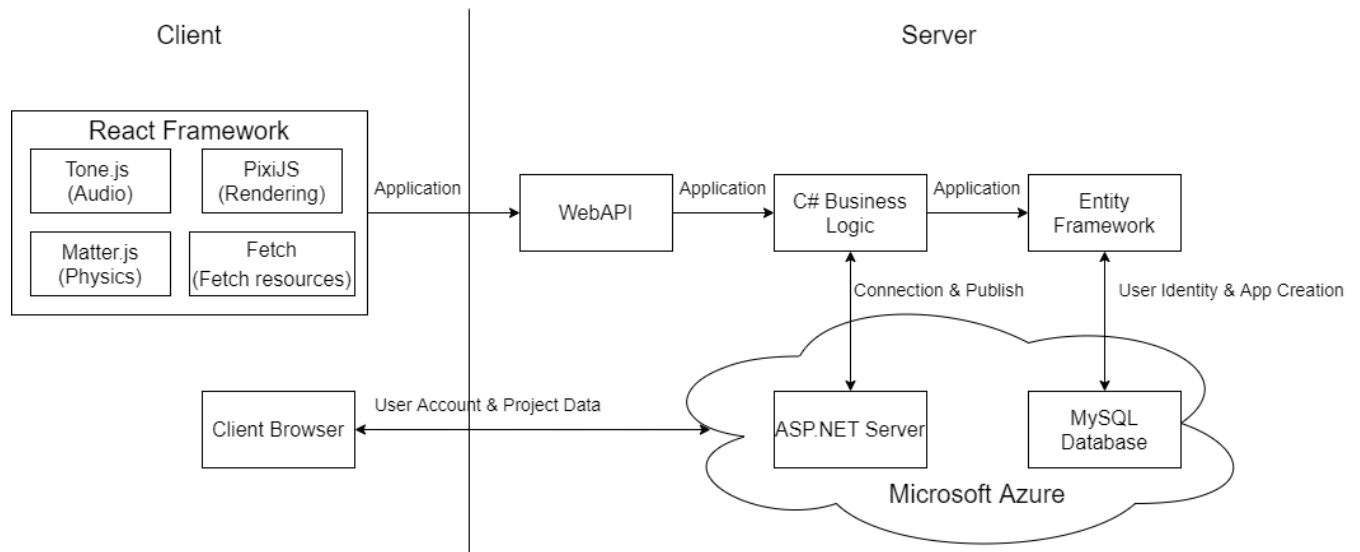
Requirements Analysis

System Architecture

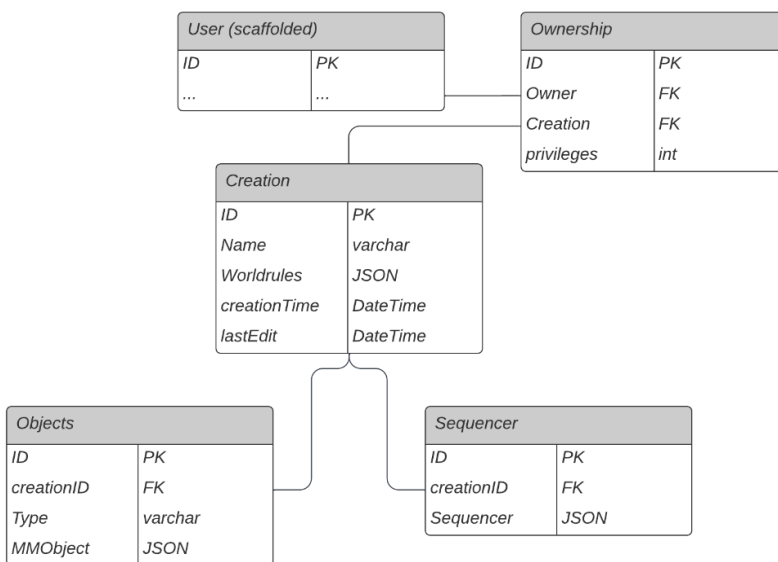Figure 0: System architecture sketch

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

B1. Rhythm Sequencing
B2. Cannon Manipulation
- Selection and moving
B3. Note Block Manipulation
- Selection and moving
B4. Object Interactions
B5. Hosted web page
B6. Component Pallet
B7. User Accounts
B8. Project Storage

Rank 2: Planned Features

P1. Project Exporting / Sharing

P2.   Animations coupled with sound

P3.   Advanced timing / trajectory control

Rank 3: Bells and Whistles

W1.   Advanced physics control

W2.   Many prebuilt instruments/sounds

W3.   Project Layers
- Change layering

Timeline

Label - Item	Due Date	Responsible Person
Design Doc Update	2022-9-14	Xiaodi
B6 - Component Pallet	2022-9-19	Jackson
B7 - User Accounts	2022-9-19	Xiaodi
B8 - Project Storage	2022-9-19	Isaac
Alpha Demo	2022-9-26	Jackson
P1 - Project Exporting / Sharing	2022-10-3	Xiaodi
P2 - Animations coupled with sound	2022-10-3	Carter
User Guide (1st draft)	2022-10-21	Jackson
Design Document update	2022-10-21	Xiaodi
Beta Demo	2022-10-28	Isaac
P3 - Advanced timing / trajectory control	2022-10-30	Isaac
W1 - Advanced physics control	2022-11-21	Isaac
W2 - Many prebuilt instruments/sounds	2022-11-28	Xiaodi
W3 - Project Layers	2022-11-28	Carter
Final User Guide	2022-12-02	Carter
Final Design Doc	2022-12-02	Xiaodi
Final Demo Day	2022-12-09	Jackson

Appendix A: UI Sketches
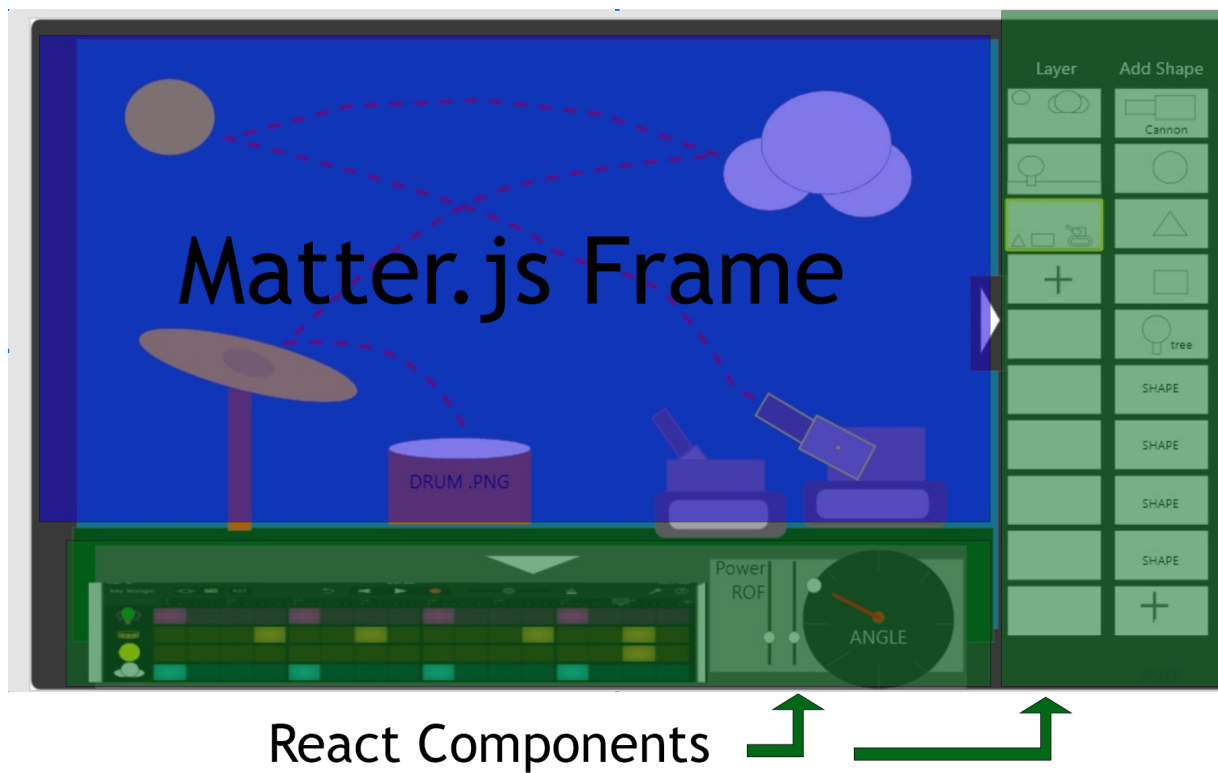
Figure 2: 2D cannon scene sketch for music tool

Figure 3: Login and sign up page

MusicTool

Welcome, aaa@gmail.com

Logout

My Projects

Project Name	Project ID	Project Workspace	Share Project	Delete Project
TestCreation	1	Go to Project	My Project	Delete

Get My Projects

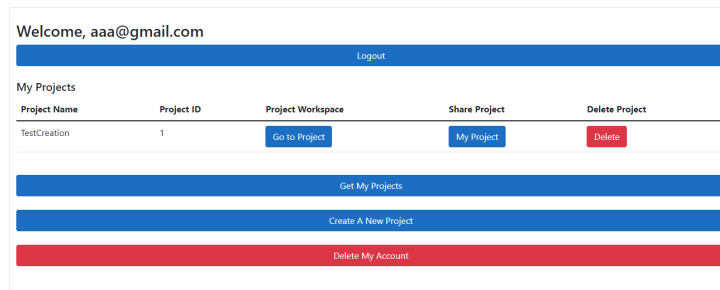Create A New Project

Delete My Account
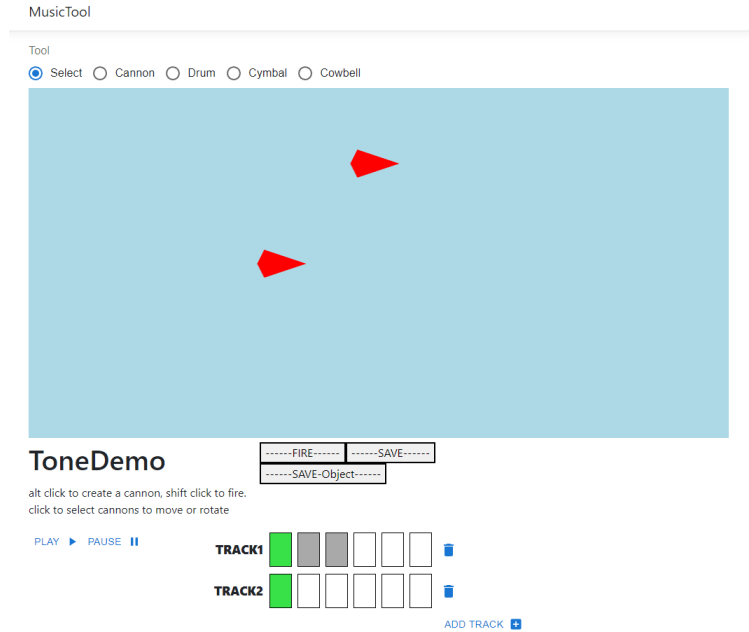
Figure 4: User Account page
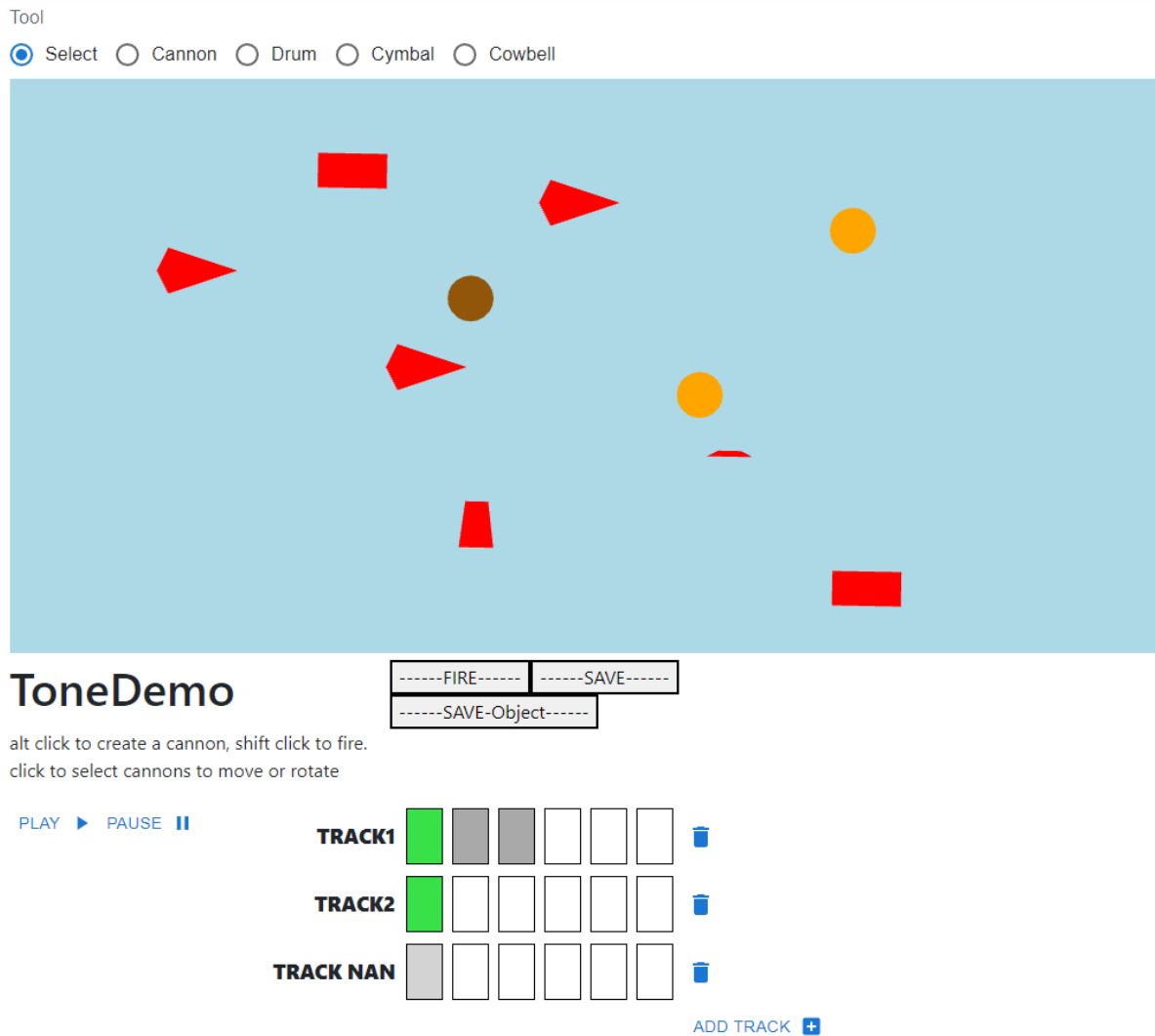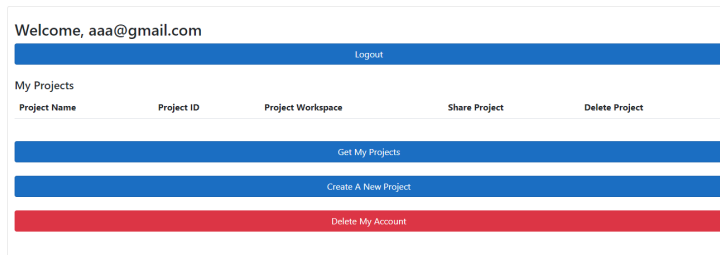
Figure 5: Combination of audio demo and cannon demo

Appendix B: Use Cases

Use Case Number	1
Title	Login/Sign up 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 Account page
Description	User want to modify the account and the list of projects.
Steps	1. After users successfully login or sign up, they should see their user account page.

2. Users click the "Get My Projects" button, then the table lists all the projects created by each user.

3. Users click the "Go to Project" button to go to their project workspace.

4. Users click the "My Project" button to share their project link to their Facebook or Twitter.

5. Users click the "Delete" button and select "Yes" to delete their project.
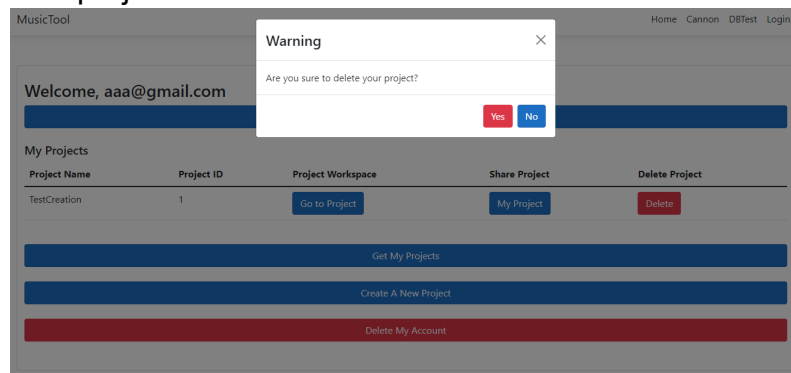
6. Users click the "Create a New Project" button to a new project workspace, then users can work on their new projects.

MusicTool

Tool

○ Select ○ Cannon ○ Drum ○ Cymbal ○ Cowbell

ToneDemo

alt click to create a cannon, shift click to fire.
click to select cannons to move or rotate

------FIRE------ ------SAVE------
------SAVE-Object------

PLAY ▶ PAUSE ❚❚ TRACK1

TRACK2

ADD TRACK ➕

7. Users click the "Logout" to go back to the Login/Sign up page.
8. Users click the "Delete my Account button" and select "Yes" to delete their account.

Related Figure	Figure 4

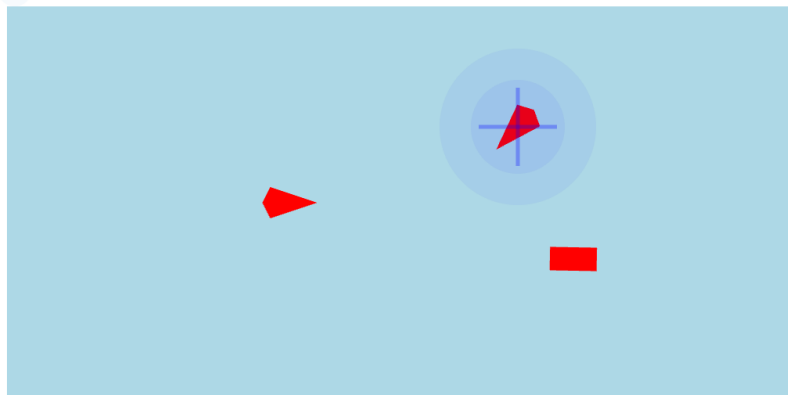| Use Case Number	3
Title	User play with cannon to shoot balls and make sounds
Description	User want to set cannons and let them shoot balls and make sounds when a collision happen
Steps	1. Users select tool and click on the board to add the

instrument
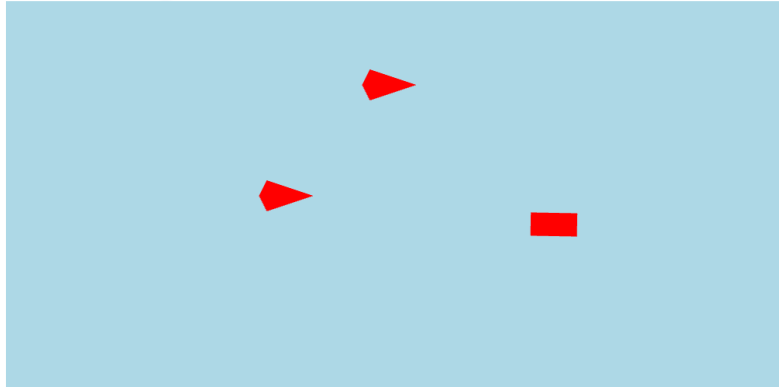
2. Users click on cannon and hold on mouse to move the cannon
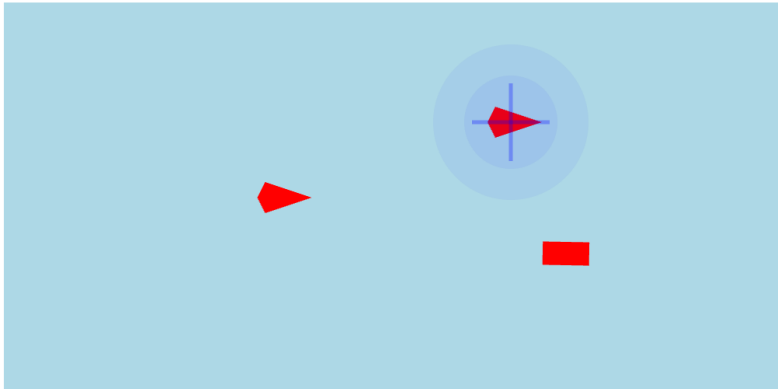
3. Users click on cannon to select it, then click on the outside circle and hold on mouse to change the angle of the cannon

4. Users select nodes on sequencer

PLAY ▶ PAUSE ❚❚

TRACK1 🟩 ⬜ ⬜ ⬜ ⬜ ⬜ 🗑

TRACK2 🟩 ⬜ ⬜ ⬜ ⬜ ⬜ 🗑

5. Users hit play, then each time when a ball is shot by a cannon, the corresponding node in sequencer makes sound.

Tool
⦿ Select ○ Cannon ○ Drum ○ Cymbal ○ Cowbell

ToneDemo

alt click to create a cannon, shift click to fire.
click to select cannons to move or rotate

PLAY ▶ PAUSE ❚❚

------FIRE------ ------SAVE------
------SAVE-Object------

TRACK1 ⬜ ⬜ 🟩 ⬜ ⬜ ⬜ 🗑

TRACK2 ⬜ ⬜ ⬜ ⬜ ⬜ ⬜ 🗑

ADD TRACK ➕

6. The melody made by sequence plays in a loop

| Related Figure | Figure 5 |