Ball Dance: A Web-based Interactive Music System

Kaushal Sali*
Jiawen Huang*
ksali3@gatech.edu
jhuang448@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

ABSTRACT

Ball dance is an interactive music system. It is implemented as a graphical web based game. This paper describes the workings of the system, its implementation details and discusses it's strengths and weaknesses.

1 CONCEPT

Ball Dance is a system intended for a user of any age group. This system is not a performance instrument but more of an interactive art piece. It can also be viewed as a generative music system that uses chaos as the generative element.

2 DESCRIPTION

2.1 Game Mechanics and Mappings

The system consists of multiple balls that float around the space and bounce of each other. There are two main types of balls: Trigger balls and Regular balls. Trigger balls are responsible for making or triggering the sounds. These balls are associated with different instruments in Ableton. There are three types of trigger balls each having a different sound design. Regular balls hold the pitch information. When a trigger ball hits a Regular ball the pitch associated with the Regular ball is played through the instrument associated to the trigger ball. The radius of the ball is used to manipulate the velocity of the midi note. All balls have fixed lifespans. As their life decreases, the start to wobble and eventually explode and die. Death of a balls makes a high pitched sound.

2.2 User interaction

There are multiple ways that user can interact with the system. Actions that can be performed by the user are:

- **Adding balls:** The user can add new balls by pressing keyboard keys. Number Keys 1, 2 and 3 let user add the Trigger balls associated to the 3 sounds. key '1': Low sound, key '2': Mid sound, key '3': High sound. Regular balls can be added by pressing key 'R'. As of now the user doesn't have control on which pitch will be associated to the newly created ball. All balls are added at the position of the mouse pointer on the screen.
- **Removing balls:** The user can remove balls by clicking on them. When clicked, the balls explode and play a sound.
- **Adding Force:** Pressing key 'F' adds random forces to all the balls and increases the chaos in the system. The Trigger balls can be steered as a group by using the arrow keys. A fixed amount of force is applied to them along the direction of the pressed arrow key.
- **Stopping:** The user can stop all motion and bring the balls to rest by pressing 'Spacebar'.
- **Looping:** By pressing key 'L' the user can keep the system in constant motion, thereby maintaining the chaos and the collisions. This keep it in a state of continuous generative system based on chaos.

3 CONTEXT

Ball Dance is a web-based implementation that users can interact it with keyboard and mouse. It is currently host locally but has the potential to be host by a server that can be accessed from any browser.

There is much related work in this space in the aspect of 2D graphical interface and physical simulation, such as *BallDroppings*[5], *Electroplankton*[6], *Musyc*[2], etc. In these systems, the sound is triggered by collision between objects, which is same in ours.

Ball Dance is different from the previous systems in the following aspects:

- **Gravity:** All of the three systems have gravity in the space. In *Ball Dance*, without gravity, the balls are moving with more freedom. And it makes sense to project it to a horizontal surface for exhibition, such as a table.
- **Simplicity:** *Ball Dance* is designed to be a minimalism system. The only object type is the ball, and the ways to trigger the sound is straight forward.
- **Lifespan:** Each ball has a lifespan assigned to it. The balls will shake when they are getting closer to death, and the shaking and exploding add more probability to the sound.

4 IMPLEMENTATION

Our system is a website, implemented using p5.js [4] for graphics and matter.js[1] as a physics engine. We use Max to interface between Ableton and the website.

4.1 Graphics

All the graphics are handled using p5.js. Every collision causes a change in the color scheme. Balls have trails drawn behind them that trace their motion. This is done by maintaining history of their motion.

4.2 Physics

matter.js [1] is a javascript based physics engine which simulates the physical interaction between bodies. We use matter.js to simulate

*Both authors contributed equally to this project.

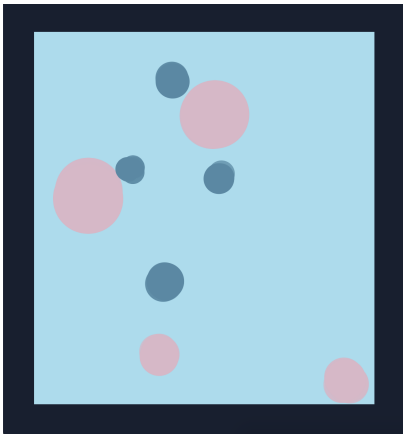

Figure 1: Interface of *Ball Dance*

and detect the ball collisions. The gravity and friction of surfaces in the physics engine is set to zero. The coefficient of restitution is set to 1 for the balls so that there is no energy loss when they hit each other. This is important to maintain a fluid movement.

4.3 Game Implementation

Ball classes describe the behiviour and lifespan of balls. All the balls are controlled by a Ball System class which maintains and manages the adding, removing and interaction of the balls.

4.4 Max/MSP and Ableton Live

The sounds are synthesized from Ableton Live. To receive message from the web app, Max/MSP is used as a bridge that receives the trigger message, decodes and maps the values to a reasonable range, and control the instrument in Ableton.

4.5 oscWebSocket

We are using WebSocket to communicate, that is, to trigger sounds in Ableton Live by sending a message from P5.js to the Max Patch. We first tried to send through UDP but it turns out it does not work for our situation. It tooks a while to figure out and we finally built the bridge based on an open-sourced repository[1]. Although there should be some way to control Ableton directly from the web app, the Max approach is easier to implement.

5 STRENGTHS AND WEAKNESSES

5.1 Strengths

The mechanism is easy to understand and the input is simple and reliable. By pressing a key, the reaction is shown on the screen immediately. The relation ship between action and sound is obvious. It is nature for user to build the connection that collision produces sound, and bigger objects produces louder sounds.

Given the fact that *Ball Dance* is a web application, it can be host by a server and accessed from anywhere with internet connection and it is independent on operation system. And the size of the playground is adaptable.

[1]https://github.com/avantcontra/osc-websocket-example

5.2 Weaknesses

Due to some inherent problems in Matter.js implementation, the collision detection is not robust. Some collisions are not detected, which leads to no sound output even though a collision is seen.

Since all sounds are triggered by a physical simulation system, users do not have control over the structure. And control over harmonic and timbre is limited. There is the option of adding buttons and sliders at the side of the playground, but this can ruin our graphical design.

Another consequence of a physical simulation system is that it can easily get chaotic. If the user continues to add balls to the space, a tiny force will cause the domino effect.

This project turns out to be much visual-focused. We have the idea of enhancing the interacting experience by visual effect. We consider it as a success in this aspect. The bouncing balls and flashing colors make the system engaging. As a result, however, we did not spend much time as we expect on the sound design part. The variety in timbre could be wider.

6 FUTURE WORK

There is a lot of scope to improve this system. Some of the things that can be done are: Provide more control over which pitches and instruments can be added to the system so as to increase the high ceiling. Color coding to differentiate the types of trigger balls will be useful and intuitive. Replacing circle shapes of balls by sprites of Gaussian distribution to give the balls a smooth smokey effect will make the visuals more appealing. Some onboarding like a help box that describes the game controls would be useful for a new user. Although Max and Ableton provide a lot of easy and variety of sounds it is not extensible as a general user website. We can implement the audio using Tone.js[3] which can help us to use it a standalone website.

REFERENCES

[1] [n.d.]. . https://github.com/liabru/matter-js
[2] Fingerlab. 2001. *Musyc.* Retrieved December 10, 2019 from http://fingerlab.net/portfolio/musyc
[3] Yotam Mann. 2015. *Interactive music with tone. js.* Citeseer. https://tonejs.github.io
[4] Lauren McCarthy. 2015. p5. js. *URL https://p5js. org* (2015).
[5] Josh Nimoy. 2003. *BallDroppings.* Retrieved December 10, 2019 from https://github.com/jtnimoy/BallDroppings
[6] Nintendo. 2005. *ELECTROPLANKTON.* Retrieved December 10, 2019 from https://www.nintendo.com/consumer/gameslist/manuals/DS_Electroplankton.pdf