Usage of Physics Engines for UI Design in NexusUI

Chase Mitchusson
Experimental Music & Digital Media
Louisiana State University
cmitc79@lsu.edu

Anthony T. Marasco
Experimental Music & Digital Media
Louisiana State University
amarasco@lsu.edu

Jesse Allison
Experimental Music & Digital Media
Louisiana State University
jtallison@lsu.edu

ABSTRACT

In preparation to expand the experimental interfaces in NexusUI widgets, the authors have been evaluating physics engines and exploring physics-based user interfaces on the web. Tying physics simulation events, influenced by user interactions, to web audio encourages exploration of novel methods of interactivity between users and web-based instruments. Object collisions, deformation of a mesh of objects with elastic connections, and liquid simulation via particle generation were identified as systems with dynamics that may provide interesting links to audio synthesis. Two popular physics engines explored are LiquidFun and Matter.js, with new prototype widgets taking advantage of LiquidFun's Elastic Particles and Matter.js' Cloth and Newton's Cradle composites. One of our goals is to discover methods of audio synthesis that complement the behaviors of each physical simulation.

1. BRINGING PHYSICS ENGINES TO NEXUSUI

The NexusUI library of user interface objects arose from work in distributed performance systems utilizing web browsers on a variety of devices.[4][6][8] The library developed with a standard core set of interfaces such as sliders, buttons and dials, but also facilitated exploratory interface design by providing a standard UI design framework and easily adaptable utility functions. Various experimental interfaces took advantage of recent additions to web browser interaction such as accelerometer data, gestural touch controls, multi-touch information, as well as simple automation – animated sliders, 2D position, the game of life simulation, and gesture recording and playback.[7] To explore animated user interfaces further required extending the animation code with physics simulations or pairing the UI with an existing physics engine.

The development of a series of widgets that play to the strength of a particular simulation, allowed for exploration into the benefits and drawbacks of the applications of each physics engine for mobile music performance.[2] Each engine's abilities to handle simulated environmental properties such as gravity, weight, elasticity, and collision handling were tested and mapped to complementary audio engine parameters and synthesis types.

1.1 Interface 1 - Newton's Cradle

The Newton's Cradle widget (built using Matter.js[3]) is comprised of a series of weighted bodies suspended from an invisible, static boundary by rigid constraints. Pulling weights away from their resting position allows them to swing back and collide with each other upon their release. After assigning a single pitch to each weight, Matter's Events module is used to monitor collision pairs and to compare the speeds of each swinging weight.

Upon impact between weights, the widget maps the pitch of the faster weight to a synthesizer built in Tone.js and triggers its attack.[5] The corresponding weight's velocity data is also monitored and mapped to the sustain and release parameters of the synthesizer's envelope. This widget has proven to be useful for performance in AM and FM synthesis due to the variety of physics data that can obtained and mapped to a synthesizers modulation parameters; in FM synthesis for example, the velocity or speed of each weight can tied to a synthesizer's harmonicity ratio to produce more bell-like tone colors for pitches assigned to faster swinging weights.

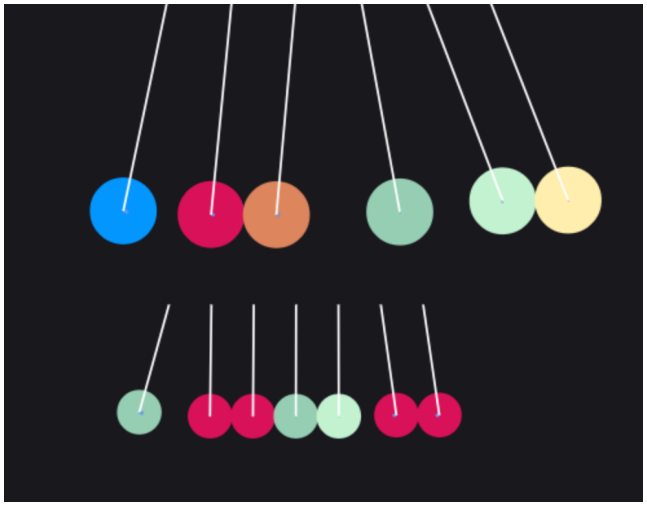

Figure 1: Example of the Newton's cradle widget in motion.

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

1.2 Interface 2 - Cloth Mesh Displacement

A suitable cloth-like structure can be built in Matter.js through a mesh of soft-body particles with elastic connections. As rigid bodies are moved into contact with the cloth, it distorts and the uniform positioning between bodies contracts and expands. The positions can be mapped directly to additive synthesis parameters such as pitch and volume. A more general set of inputs can be derived by monitoring the change in uniformity across the cloth and mapping to synthesis parameters such as the relative frequency of overtones, effectively controlling the harmonicity of the synthesis. These distortions from a resting state can be mapped to sound in other ways such as playback speed or position in an audio buffer, pitch shifting, and the shifting of FFT bins.

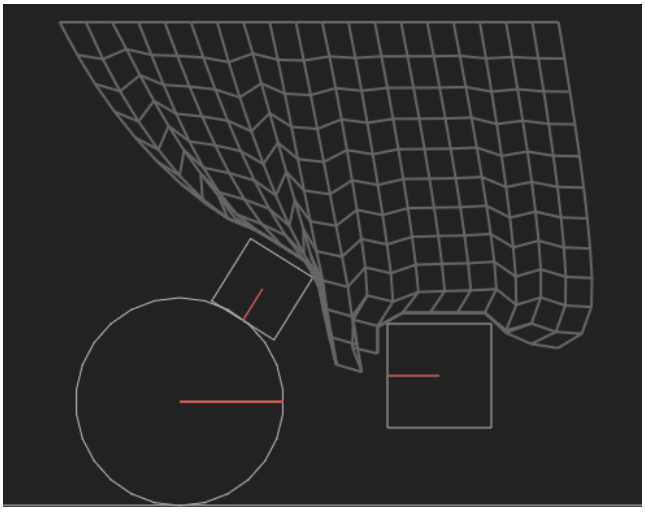

Figure 2: Example of Cloth being draped over several rigid bodies while reacting to simulated wind and gravity elements

1.3 Interface 3 - Elastic Particles

The Elastic Particle tool contains three particle groups and a rigid circle body. Dragging the rigid circle body into the particle groups smashes and squishes the particle groups. The interactions in Elastic Particles were initially planned to distort sound based on the change in shape of the particle groups, however, the particle system for the JavaScript port of LiquidFun is problematic for detecting collisions.[1] Since the particle collision data is inaccessible, data useful for musical mapping has to be found by manipulating position and velocity buffers. Synthesizer parameters in Tone.js are mapped to the delta value of the average velocity of the particles, which are then interpolated through a preset array. The position data is mapped to a vibrato effect, which intensifies as the particles approach the top of the screen.

2. CONCLUSIONS AND FURTHER DIRECTIONS

Following the design and testing of these initial widget prototypes, both libraries offer unique potential for further usage. The Matter.js physics library provides the most robust set of data points and event callbacks for mapping to audio synthesis parameters. LiquidFun was intriguing especially for its particle generation system, however, the most current release of LiquidFun's JavaScript port is missing bindings for gathering particle data that are found in the C/C++ libraries. For incorporating into NexusUI widgets, a custom JavaScript port would need to be built and maintained severely limiting its usability.

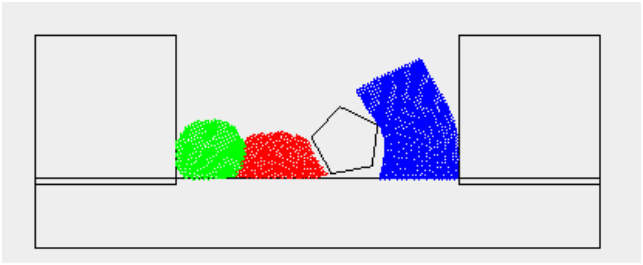

Figure 3: Example of Elastic Particles reacting to the force and pressure exerted upon them by a rigid body

As expected, mapping of the interactions facilitated by the physics simulations to audio synthesizers varied widely in terms of usability and effective sonic output. However, the added layer between interaction and control values - e.g. interaction with a virtual physical object and having that object's influence in the virtual world create control values, made for a more complex mapping and interaction in general. Using the interface oftentimes became engaging whether or not the sonic output was useful. When the two reinforced each other, interesting interaction with a physics simulation and interesting sonic output, exploration of the instrument became quite rewarding.

Avenues for development and exploration are plentiful: creating composite machines and structures that simulate different systems, creating different synthesis engines that pair with the salient features of these physics simulations, and user studies elucidating the dynamics between user and virtual physical systems in terms of performability, intuitive interactions, and affordances for performance.

3. REFERENCES

[1] Liquidfun - http://google.github.io/liquidfun/documentation.

[2] LSU EMDM: nx Physics UI - https://github.com/lsu-emdm/nx-physicsui.

[3] Matter.js - http://brm.io/matter-js/.

[4] Nexus User Interface - http://nexusosc.com.

[5] Tone.js - https://tonejs.github.io/.

[6] J. Allison, Y. Oh, and B. Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. In *Proceedings of the New Interfaces for Musical Expression conference*, 2013.

[7] B. Taylor and J. T. Allison. Gesture capture, processing, and asynchronous playback within web audio instruments. In *ICMC*, 2015.

[8] B. Taylor, J. T. Allison, W. Conlin, Y. Oh, and D. Holmes. Simplified expressive mobile development with nexusui, nexusup, and nexusdrop. In *NIME*, pages 257–262, 2014.