ThereminFreaks – Theremin Rhythm Game

Group 34
Esteban Looser-Rojas
Daniel Olivas Hernandez
Michael Recklein
Introduction

- Desire to make a video game with unique hardware
- No rhythm games out there for simulating theremin
- Very unique instrument to make a rhythm game for: no contact + wave hands around
Objectives

- Create realistic and intuitive theremin simulator
- Responsive and enjoyable game
- Sound engine sounds like theremin and controls like one
Block Diagram

**Theremin**
- Volume Sensor
- Pitch Sensor
  - Modulated sine waves
  - Frequency detection
  - Analog frequency values
  - Analog to Digital Conversion
  - Frequency samples
  - Microcontroller
  - USB to RS-232

**PC-Side**
- Graphics Engine
  - Render Instructions
- Audio Engine
  - Song & theremin controls
- Game Logic
  - 5 V @ .5 A
  - USB
  - Device Driver
  - Theremin controls
  - Power
  - Data
Pitch and Volume Sensors

- Volume antenna: 6” wide; 10” long; 3/8” dia
  Pitch antenna: 2’ long; 3/8” dia
- Design taken from DIY theremin guide
- Problem: unstable oscillators
- Theory: antennae load too large
- Solution: shorter antennae; better oscillator design
Oscillators

- First used Hartley oscillators w/SMD transformer – very weak signal
- Change: Hartley → Colpitts (thru-hole inductors)
- Requirement: Antennae change osc. freq by 5-10 kHz from base freq
- Measured antenna freq from no hand near (left) to hand near (right): ~6 kHz difference
Old design: Hartley Oscillator

New design: Colpitts Oscillator
Mixers

- Initially used Gilbert cells (on left) – sensitive; low output (1.7V pk-pk)
- Change: oscillator sine wave → square wave
- Requirement: modulated waves from 1-4 MHz input at logical high (2-5V)
- Use XOR gate to mix signals (on right) – get clean 5V output
Gilbert Cell

Square Wave & Digital XOR Mixer
Detector and Integrator

- Use BJTs for integrator circuit
- Simple diode and RC lowpass filter for detector
- Requirement: 30 kHz bandwidth
- Ran func. generator through 20 Hz - 30 kHz range
Integrator and Detector

Above: 50 Hz input
Below: 2 kHz input
Analog Circuit

From antenna

Oscillators

Mixer

To digital section

Integrator

Detector
Game Design: Rendering Engine

- Create mesh to signal required pitch and volume
- Generated from text file + primitive meshes
- Requirement: want ~60 fps consistently – achieved (checked FPS counter)
Game Design: Game Logic

- Decoupled frame rate from game speed
- Scoring function based off cubic distance from “safe zone”
- Forgives shaky hands and sampling noise
Game Flowchart

Start

Instantiate Graphics Engine

Instantiate Audio Engine

Instantiate Local Variables

Generate Song Mesh

Reset Score Trackers

Load and Render Song Scene

Beginning of Time Step

Reset Score Multiplier and tracker

If: Correct Note Is Played

Add to Score Total: Base x Multiplier

Increment Multiplier Tracker

If: Song is Over

Display Score and Ranking

Return to Song Selection Menu

Unload Song Scene
Audio Engine

- Take ADC samples from driver and scale to a certain frequency range
- Analyzed recorded theremin sound with FFT (left)
- Simulate theremin sound using additive synthesis (right)
Audio Engine cont.

- Requirement: have at least four-octave sound range
- Used 110 Hz as lowest note (A2, left) and 1710 Hz as highest note (A6, right)
- Five-octave sound range result
Synthesis Flowchart

Start

Set up STK at 44.1 kHz sample rate

Set fundamental frequency

for i = 0; i++

overtones[i] = freq * (i + 2)

Receive ADC samples

Add all partials together

Output audio

If vol sample changed

true

Attenuate partials to vol voltage

false

If pitch sample changed

true

Scale frequencies to pitch voltage

false

for i = 0; i++

overtones[i] = freq * (i + 2)
# Analog-to-Digital Converters

- Off-the-shelf 16-bit Maxim Integrated σ-δ converters

## Requirement

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCs can send at least 480 samples per second</td>
</tr>
</tbody>
</table>

## Verification

<table>
<thead>
<tr>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect trimmer to ADC; ADC to Arduino</td>
</tr>
<tr>
<td>Program Arduino to manipulate ADC and send voltage reading</td>
</tr>
<tr>
<td>Verify trimmer position corresponds to value from ADC</td>
</tr>
</tbody>
</table>
PIC16 and RS-232 to USB chip

- PIC16 signals to ADC to take sample
- PIC16 then sends four bytes to PC thru RS-232 UART
- Requirement: PIC16 able to send samples from ADC to RS-232 interface at 9600 baud
- Verification: Send byte to PIC16, get same byte back + both ADC samples
Oscilloscope Trace of RS-232 Transmission
Digital Circuit
USB to RS-232
PIC16
Integrator
Mixer
Oscillators
ADCs
Uses system calls to open serial device and read/write from/to theremin controller

Requirement: provide 2 16-bit samples from theremin with delay < 50 ms

Verification: take time difference between PIC sampling input and driver receiving input

~9.5 ms delay
Conclusions

- Oscillators still need work done (detailed in next slide)
- Digital section of theremin reliable
- Synthesis engine decent
- Game barebones but working
Future Work

- Fix transistor biasing with inductors before biasing inductor 6V pk-pk (left); 32.4V pk-pk after (right)
- Improve upon sound synthesis engine
- Flesh out game
- Actual controller enclosure and rigid antennae