1. Introduction
The Whatcha-ma-Thumper is an interactive mechanical percussionist, which responds to timing of the user waving their hands over light sensors by playing various instruments at a related tempo. There are four instruments: a drum, a triangle, a clapper, and a strummer. Our concept was to create a musical toy that would engage the user both visually and through sound, combining familiar elements in an unfamiliar way.

2. Mechanical elements
Each instrument has a mechanical apparatus that is mostly separate from those of the other instruments, which allowed for parallel and independent development and testing. The goal of these separate units is to make musical sounds from their associated
instruments. The following sections describe the mechanics of each of these player modules.

2.1. The Drum
The goal of the drummer module is to strike the drum once when given the appropriate signal from the Cricket. There are three main mechanical components that accomplish this task: the hammer, the cam, and the motor.

The hammer
The drum is struck by a hammer that is mounted on an axle. This allows one degree of freedom: the head of the hammer may move up or down. The resting position comes from the balance between gravity (pulling the head down) and a spring opposite the head. If the hammer head is lifted (or the tail is pushed down) and then released, the weight of the head swings it down until the tail contacts the spring. The head bounces a few times until it comes to rest.

The ideal operation is for the head to strike the drum only once upon the initial bounce. The spring, which is used to counteract the downward motion of the hammer, was made adjustable for this reason. The spring is mounted in such a way that it can be raised or lowered slightly to accommodate the height of the drum, allowing this motion to be fine tuned.

The cam
In order to cause the hammer head to be lifted and dropped using a simple motor, a snail cam is mounted on an axle such that it can press down on the tail of the hammer. As the cam turns it pushes the tail down, lifting the hammer’s head, until the cam’s tooth passes the end of the tail, at which point the tail is released and gravity causes the head to drop back down.

One unanticipated issue in our design was that the cam was slightly too close to the hammer's tail. As the tail’s resting position was raised (by adjusting the spring), the torque required of the cam to move the tail increased. The adjustability of the spring unexpectedly affects the cam's operation, and not just the resting position of the hammer head. Due to this problem, the resting position of the hammer head is now much more constrained that we had initially hoped. Fortunately, this did not prove to be a major problem, since we were able to adjust the height of the drum itself to accommodate the limited range of motion.
The motor
We had initially attempted to mount the motor at the base of the hammer mechanism, between and below the two main axles, using a rubber band as a friction belt to drive the snail cam. Unfortunately, the rubber band pulled the cam's axle too tightly against the mounting boards, increasing friction on the axel to the point where the rather weak motor could no longer overcome it.

Eventually we decided to do away with this extra mechanical problem by making the motor drive the cam axle directly. By making the motor coaxial with the cam, the friction placed on the axel was reduced, and the motor was able to drive the cam without difficulty.

2.2. The Clapper
The clapper instrument consists of two broad flat pieces of wood (a.k.a. wings) that make a percussive noise when slapped together. To facilitate this, one wing is stationary, while the other can rotate a few degrees around an axle. The mechanical player thus needs to move the mobile wing away from the stationary wing, then quickly slap it back to create the clapping sound. Apart from the mobile and stationary wings, the mechanical components which execute a clap are a cam, a motor, and a rubber band belt connecting them.

The mobile wing
The flat, wooden plane of the mobile wing is connected to a curve copper tube, at the end of which are a set of lead weights. The middle of the tube is mounted on a wooden disk, which has a central hole through which the axel passes. The lead weights on the end of the tube act as a counterbalance to the weight of the wooden wing, so that in its resting position the mobile wing is pressed against the fixed wing.

The cam
A snail cam is used to press out and up on the mobile wing, so that the two wings are drawn apart. As the cam passes its peak, the counterbalance weights cause the mobile wing to snap back to its resting position against the fixed wing. This is the same basic design that is utilized by the drum player.

The motor
The motor that drives the clapper is attached to the framework below the entire clapping mechanism. A rubber band provided a friction belt between the axles of motor and cam.
2.3. The Triangle
The triangle is a simple suspended metal triangle, with one corner disconnected to allow reverberations. The mechanical playing mechanism strikes the triangle with small metal mallets.

These mallets are attached to a wooden drum which, when turned via a rubber band belt (similar to the arrangement used for the clapper), strike the triangle. The triangle is suspended from the top of the automaton by thin elastic bands, allowing the triangle to resonate when struck, but holding it in place so that it can reliably be struck again.

2.4. The Strummer
The strummer is made from a taught elastic string attached to a metal can which acts as a resonator. When the string is plucked, a light twang emerges from the can.

The strummer is plucked by a series of plastic fins which are mounted on a wooden drum. The drum itself is connected by a rubber belt to a motor, so that when the motor turns, the drum rotates, and the fins pluck the string.

3. Computational and Electronic Elements
In addition to the motors, which drive the instrument player modules, the other electronic components include the crickets for computation and sensors for interaction and feedback.

3.1. User interaction sensors
Rather than using switches that rely on physical contact, three photocells were used to allow the user to control the actions of the instruments. The user can trigger changes in tempo by waving his or her hands over the copper tubes in which the light sensors are embedded. The Crickets detect this interaction by storing an initial ambient light reading when they are first turned on, then watching for the sensor readings to decrease by a fixed
threshold. Provided that the ambient light in the room is mostly directed down from the ceiling, we found that this technique provided very reliable readings under a variety of light levels.

3.2. Feedback sensors
Reed switches are used to provide feedback to the Crickets regarding the position of three of the four instruments. (The drummer was excluded due to the limited number of sensor ports that we had available.) Magnets are strategically attached to different places on the cams or drums for each instrument in such a way that the reed switches can sense when each magnet has passed, so that each Cricket is able to sense when to turn off its motor. In the case of the clapper and triangle, this corresponds to one slap or chime. For the strummer, this corresponds to two plucks.

This feedback was necessary due to the very limited control that the Crickets have over the motors. Unlike servos, the Lego motors cannot be directed to rotate a fixed number of degrees at a set speed, so we needed some objective way to judge when to turn off each motor in order to prevent additional, accidental beats from the instruments. Rather than relying on guesswork to turn a motor on or off for a preset window of time, we were able to control the rotation of the motors with a useful amount of accuracy.

3.3. Computation
The computational components of this system are implemented in Cricket Logo, and executed by three Cricket nodes. The Crickets act independently, with no communication amongst themselves. This limits the sophistication of the system since each instrument is controlled separately, but in practice we felt that there were mechanical issues that would have made such interconnections fruitless anyway (see Section 6).

The actual programs that control the instruments are quite simple. Each cricket reads from two sensors: one reed switch to sense the position of each instrument, and one photocell to sense user interaction. A state machine rotates between sensing a tempo and halting the instrument, and this state dictates when the motor should turn on.
The code for each Cricket is similar in structure, but customized for the particular instrument it controls. The interaction of the reed switches, the driving motors, and the mechanics of each instrument is unique enough to require this. These differences basically translate into slightly different behavior of each Cricket in order to give the correct timing for the instrument.

4. Process of Construction
Once the basic design was in place, the first steps in construction were to create the copper tubing framework that serves as the backbone of the piece. Apart from the drummer and clapper wings, the other mechanical components were fairly standard. With this framework in place, we began creating the instruments and player modules separately.

4.1. The Drummer
The drummer framework is mostly composed of laser-cut wood. After creating hand-drawn conceptual designs, Corel Draw was used to lay out the framework to scale. The design approach was to create interlocking pieces that could be tested and individually replaced without requiring glue or other processes that would be difficult to undo. This turned out invaluable as some pieces needed to be recut several times during the development of the drummer.

The hammer, axle, support struts, and spring mount were designed and prototyped first. The expected issues were the spacing of the pieces in relation to themselves as well as the drum, and also the drum rest position.

Initially the design assumed the spring would be attached to a flat wooden piece with glue or tape, and have a fixed position. After cutting the hammer and support structure and combining those pieces, the problem of how to attach the spring was considered.

4.2. The Clapper
The clapper was constructed from a combination of wood and copper tubing. Two wings were designed in Corel Draw and printed out on the laser cutter, along with large wooden washers on which they were mounted. A snail cam was also printed in wood, which was mounted below the two wings.

There was some difficulty in getting the wings positioned properly relative to each other, and even more difficulty getting them to clap together properly. We initially attempted to use elastic bands to draw the wings together, but the Lego motors that we had available proved to be too weak for this strategy to be effective. The best solution proved to be the use of a counterbalance, which didn’t cause as much resistance as elastic, yet still allowed for a satisfying smacking sound when the snail cam released the weight.

4.3. The Strummer
Due to its size, the Strummer was the first instrument to be constructed, and its structure provided the framework for the rest of the instruments. A long length of copper tubing
was coiled up to a height of approximately two feet, then reinforced with additional tubing to provide structural stability. A metal paint can was attached to the top to serve as a kind of resonance chamber, and an elastic band was drawn through a hole in the can and tied to the base of the structure. Several different types of elastic were tried at many different tensions; we ultimately got the best sound from a length of thin, fabric covered elastic, drawn fairly tight.

Using the laser cutter, a wooden drum was created for the purpose of mounting the picks that would strum the elastic band. Pieces of stiff plastic were fixed to the edge of the drum, and the drum was then mounted on another piece of copper tubing which acts as the axel. An elastic belt connects this drum to a motor which drives the strummer.

4.4. The Triangle
The triangle was by far the simplest of all the instruments, requiring minimal time to assemble. A store-bought musical triangle was suspended from the framework that had already been constructed for the strummer, and an additional copper tube was added to provide the axel for the drum on which the mallets were mounted. The drum itself was designed with Corel Draw and cut on the laser cutter, and included a hub for the drive belt. Large nails were attached to the drum to serve as the mallets, the ends of which were bent so that they would not get caught on the triangle when they struck.

5. Aesthetics and Education
We chose an open and abstract aesthetic, rather than a narrative theme. Additionally, we wanted a feeling inspired by Dr. Seuss's artwork, particularly the wild, twisted “one man band” musical instruments that he often portrayed. For this reason, as well as for ease of construction, we chose to use copper tubing as the framework, since the material allows a great deal of flexibility in its use. The open structure of our design exposes all of the mechanical elements, as well as the sensor locations and wiring, and invites exploration of the physical interactions that cause the automaton to function.

The internal state machine that drives the actions of each Cricket leads to an interesting, if somewhat confusing, form of human interaction. We can see this as being a form of puzzle, wherein the user gradually gains an understanding of how the pieces interact, and how one’s actions cause different effects. We would also hope that the musical theme would help inspire exploration of music and rhythm, and perhaps interest into the basic nature of music itself.

6. Future Work
There are numerous features that could be added to this automaton, but future work should probably focus on some problematic issues with the current piece.

The main problems relate to instrument timing and musical quality, and the source of these problems rests largely on the nature of the motors that we had at our disposal. The Lego motors have several qualities that detracted from our ability to use them effectively: they are weak, they are slow, and they are very loud. Simply using quieter motors would
be a major improvement, since the grinding of the current motors is so loud that it tends to drown out the percussion from the instruments. Motors that are faster and more powerful would allow us to adjust the timing more effectively, so that the instruments could be played as rapidly and forcefully as needed. We feel that more sophisticated programmatic control of the automaton would only be effective once these issues are resolved.

If the mechanical issues could be resolved, there seems to be great potential for adding more complex computational elements to this automaton. One of our original concepts was to invoke a kind of call-and-response sequence, in which the user would play a sequence on the instruments, and then the automaton would play back a modified version of that sequence. Networked interaction between the sensor nodes would allow for a wide range of interesting possibilities.

Finally, we would like to be able to use a different mechanism for human interaction. While light sensors provide an amusingly abstract interface, it would be preferable for the user to be able to interact with the instruments directly: hitting the drum or the triangle, closing the clapper, or plucking the strummer. This kind of interface would require much more sophisticated sensing than the kind that we are using at present, but it could provide a much more interesting experience for the user.
Appendix A. Source Code

The source code for all three Crickets follows the same basic outline, but the programs differ in the specifics of the motor operation. These differences are due to the variations in the construction of each instrument, and in particular, to the positions of the reed switches relative to the magnets.

**clapper.lgo**

```lgo
;;; Global variables
global [
  play ; Boolean: 1 = play instrument, 0 = don’t play
  initLight ; Initial measure of ambient light
  delay ; Delay between beats
  delayState ; Current internal state of the instrument
  break ; Needed to help with the state machine
]

;;; Initialize global variables, start the thread that listens to
;;; the photocells, and start the main event loop.
to main
  setinitLight sensorb ; Detect the initial ambient light
  setdelayState 2
  setdelay 50
  setplay 0

  ; Create a thread to listen to the light sensors. The
  ; internal state of the instrument will change when the
  ; sensor reading dips by 20 or more units.
  when [(initLight - sensorb) > 20] [calcDelay]
    a, setpower 6
  end

  player
end

;;; Main event loop. Plays the instrument once, then waits for
;;; the amount of time specified by the ‘delay’ global variable,
;;; then repeats. Halts if the ‘play’ Boolean is set to 0.
to player
  a, on
  if sensora > 1 ; If this is true, the reed switch is closed.
    [a, off
     wait delay
     a, onfor 2]
  if play = 0
    [a, off
     waituntil [play > 0]]
  end
end
```
;; Controls the state of the instrument. State 0 starts a timer
;; to measure the delay. State 1 sets the delay to be the difference
;; in time between the start of State 0 and the start of State 1.
;; State 2 stops the instrument from playing.
to calcDelay
  setbreak 0
  if delayState = 0
    [setdelayState 1
      setplay 1
      note 119 1
      setbreak 1
      reset]
  if delayState = 1 and not break
    [setdelayState 2
      note 88 1
      setdelay timer / 100
      setbreak 1
      send delay]
  if delayState = 2 and not break
    [setplay 0
      note 67 1
      setbreak 1
      setdelayState 0]
  end

drum_and_strum.lgo

;; Global variables
global [
  play ; Boolean: 1 = play instrument, 0 = don’t play
  initLight ; Initial measure of ambient light
  delay ; Delay between beats
  delayState ; Current internal state of the instrument
  break ; Needed to help with the state machine
  drumCount ; The drum’s motor is so slow that it needs to
              ; stay on for multiple beats. This variable counts
              ; those beats.
]

;; Initialize global variables, start the thread that listens to
;; the photocells, and start the main event loop.
to main
  setinitLight sensorb ; Detect the initial ambient light
  setdelayState 2
  setdelay 50
  setdrumCount 0
  setplay 0
  a, setpower 8
  b, setpower 4

  ; Create a thread to listen to the light sensors. The
  ; internal state of the instrument will change when the
  ; sensor reading dips by 20 or more units.
  when [(initLight - sensorb) > 20] [calcDelay]
player
end

;; Main event loop. Plays the instrument once, then waits for
;; the amount of time specified by the 'delay' global variable,
;; then repeats. Halts if the 'play' Boolean is set to 0.
;; Note that this program controls both the strummer and the
;; drum, although only the strummer is augmented with a reed switch
;; to indicate its position.
to player
  a, on
  b, on
  if sensora > 1  ; If this is true, the reed switch is closed.
    [ifelse drumCount > 4
      [a, off
        setdrumCount 0]
      [setdrumCount drumCount + 1]
    ]
  b, off
  wait delay
  a, onfor 2
  b, onfor 2]
  if play = 0
    [a, off
     b, off
     waituntil [play > 0]]
  player
end

;; Controls the state of the instrument. State 0 starts a timer
;; to measure the delay. State 1 sets the delay to be the difference
;; in time between the start of State 0 and the start of State 1.
;; State 2 stops the instrument from playing.
to calcDelay
  setbreak 0
  if delayState = 0
    [setdelayState 1
     setplay 1
     note 119 1
     setbreak 1
     resett]
  if delayState = 1 and not break
    [setdelayState 2
     note 88 1
     setdelay timer / 100
     setbreak 1
     send delay]
  if delayState = 2 and not break
    [setplay 0
     note 67 1
     setbreak 1
     setdelayState 0]
end
triangle.lgo

;; Global variables
global [ 
   play  ; Boolean: 1 = play instrument, 0 = don’t play
   initLight ; Initial measure of ambient light
   delay ; Delay between beats
   delayState ; Current internal state of the instrument
   break ; Needed to help with the state machine
]

;; Initialize global variables, start the thread that listens to
;; the photocells, and start the main event loop.
to main
   setinitLight sensorb ; Detect the initial ambient light
   setdelayState 2
   setdelay 50
   setplay 0

; Create a thread to listen to the light sensors. The
; internal state of the instrument will change when the
; sensor reading dips by 20 or more units.
when [(initLight - sensorb) > 20] [calcDelay]

player
end

;; Main event loop. Plays the instrument once, then waits for
;; the amount of time specified by the ‘delay’ global variable,
;; then repeats. Halts if the ‘play’ Boolean is set to 0.
to player
   a, setpower 3
   a, on
   if sensora > 1  ; If this is true, the reed switch is closed.
      [a, setpower 4
         a, onfor 2
         a, off
         wait delay
      ]
   if play = 0
      [a, off
         waituntil [play > 0]]
   player
end

;; Controls the state of the instrument. State 0 starts a timer
;; to measure the delay. State 1 sets the delay to be the difference
;; in time between the start of State 0 and the start of State 1.
;; State 2 stops the instrument from playing.
to calcDelay
   setbreak 0
   if delayState = 0
      [setdelayState 1
         setplay 1
         note 119 1
         setbreak 1]
if delayState = 1 and not break
    [setdelayState 2
        note 88 1
        setdelay timer / 100
        setbreak 1
        send delay]
if delayState = 2 and not break
    [setplay 0
        note 67 1
        setbreak 1
        setdelayState 0]
end