MC Squared

a musical experience

Introduction to Physical Computing

Fall 2008

Diego Rioja

Filippo Vanucci

India Amos

MC squared . . .

- is a musical instrument that detects motion and generates sounds from each side
- can be played by one or more than one person
- can be played with any part of the body, or an object
- is simple, intuitive, playful, expressive

what people are saying about MC squared

- “Sexy!”
- “This is so fun!”
- “Awesome!”

user profile

- Humans, age 7 and up. Cats, age 6 weeks and up.
- Musicians and nonmusicians.
- People who have phobias about touching things.

A cat playing a theremin—one of many such talented felines.

inspiration

- Experimenting with new ways of playing, new interactions for music performances.
- The theremin, an unusual, hands-off device for music composition.
- The Groovebox (aka the Roland MC-505), a drum machine that incorporates a motion sensor as one of its many controls.
- Somewhat anachronistically, Murat Konar's loopqoob, which Diego discovered during our observation and research phase.

Filippo showed a video clip of Jon Spencer playing a theremin. It's pretty fine.

videos: Groovebox observation

CLICK IMAGE TO PLAY

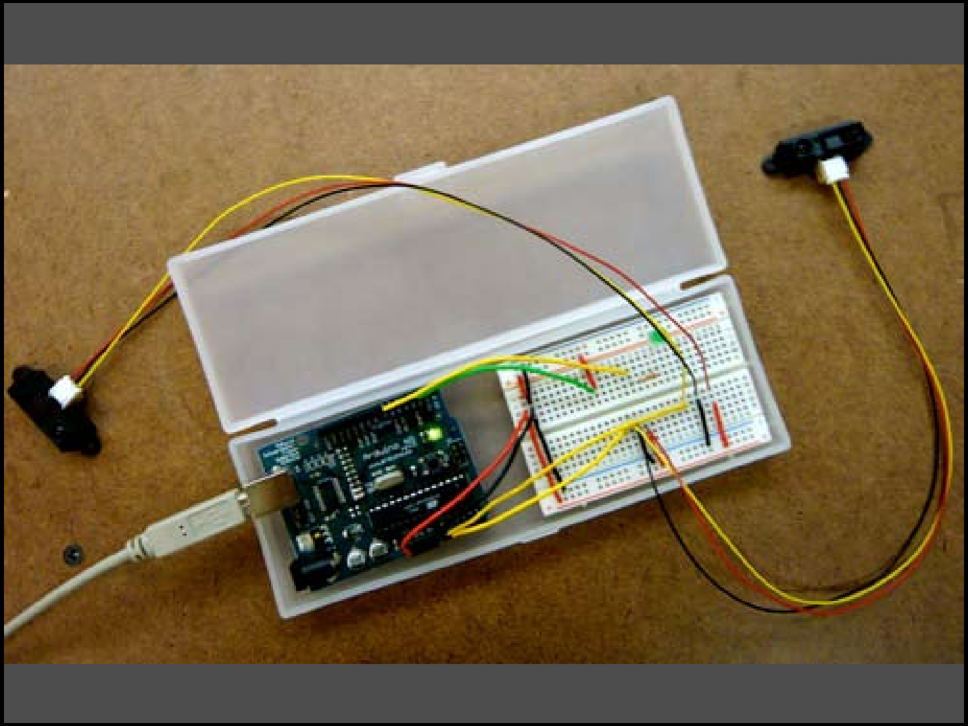
CLICK IMAGE TO PLAY

Diego set up his Groovebox in the ITP lobby, and we observed several people using it. This device was one of the inspirations behind our project. I has a motion sensor at the top, just to the left of the numbers, which generates wacky thereminic sounds.

HOW MC squared works

- Each side is embedded with an infrared sensor that detects movements within a range of about 2 to 12 inches in front of it.
- Signals from each sensor trigger a different sound, using the Minim library in Processing.

video: Initial version



CLICK IMAGE TO PLAY

Diego and FIlippo play our first prototype. This two-sensor model played sounds continuously, as long as an object was in range of the sensors. Nearer objects generated one sound from each sensor; farther objects, another, for a total of four sounds. The code is on India's blog.

video: An intermediate version

CLICK IMAGE TO PLAY

Diego plays a version with more sensors and revised code.

video: The penultimate (?) version

CLICK IMAGE TO PLAY

Carolina Vallejo, Sara Bremen, Eyal Ohana, Filippo, and Diego trying out the newest version on Tuesday evening. This model sports a new, larger foam box and plays each sound clip only once for each (contact-free) hit.

```
strates how to use the
<code>play</code> method of a
<code>Playable</code> class.
  * The class used here is
<code>AudioPlayer</code>,
but you can also play an
<code>AudioSnippet</code>.
  * Playing a
<code>Playable
it to begin pl
current positi
reaches
  * the end of
it will emit s
not stop! In c
you play somet
  * it gets to
file, it will r
rewind, it wil
try to read th
  * nothing an
to the audio s
call <code>isP
at that point,
true,
  * because the player is
still trying to read the file,
think of a record player that
gets to the end of a record.
  * It just goes around on
the same groove. It’s not
making any sound (well,
crackles maybe) but it is
still playing.
  * Press ‘p’ to play the
file.
  *
  */

import processing.serial.*;
import ddf.minim.*;
Serial myPort;
Minim minim;
AudioSample sample1;
AudioSample sample2;
AudioSample sample3;
AudioSample sample4;
AudioSample sample5;
AudioSample sample6;
int[] sensors;
int[] preVal = new int[6];
//int val=0;
int[] val = new int[6];
void setup(){
```

code

- **Processing:** mc_squared_processing_code.zip (957KB)
- **Arduino:** mc_squared_arduino_code.zip (64 KB)

```
    preVal[i]=0;
    val[i]=0;
  }
  minim = new Minim(this);
  sample1 = minim.loadSample(
“kick01.wav”, 2048 );
  if ( sample1 == null ) {

  minim = new Minim(this);
  sample4 = minim.loadSample(
“loop2.wav”, 2048 );
  if ( sample4 == null ) {
    println( “Didn’t get
kick!” );
  }
  minim = new Minim(this);
  sample5 = minim.loadSample(
“hihat.wav”, 2048 );
  if ( sample5 == null ) {
    println( “Didn’t get
kick!” );
  }
  minim = new Minim(this);
  sample6 = minim.loadSample(
“loop1.wav”, 2048 );
  if ( sample6 == null ) {
    println( “Didn’t get
kick!” );
  }
  /*  List all the available
serial ports. Don’t really
need to do this, since
   it’s always zero for me,
but it doesn’t hurt.
   */
  println(Serial.list() );
  //  If I wanted a port
```

```
  myPort = new Serial( this,
Serial.list()[2], 9600 );
  //  Read bytes into a buf-
fer until you get a line
feed.
  myPort.bufferUntil( ‘\n’ );
}

Val[2]==0){
    println(“BAM”);
    soundSample3();
  }
  if (val[3]==1 && pre-
Val[3]==0){
    println(“BAM”);
    soundSample4();
  }
  if (val[4]==1 && pre-
Val[4]==0){
    println(“BAM”);
    soundSample5();
  }

  if (val[5]==1 && pre-
Val[5]==0){
    println(“BAM”);
    soundSample6();
  }
   preVal[0]=val[0];
   preVal[1]=val[1];
   preVal[2]=val[2];
   preVal[3]=val[3];
   preVal[4]=val[4];
   preVal[5]=val[5];
}
void serialEvent( Serial
myPort )
{
```

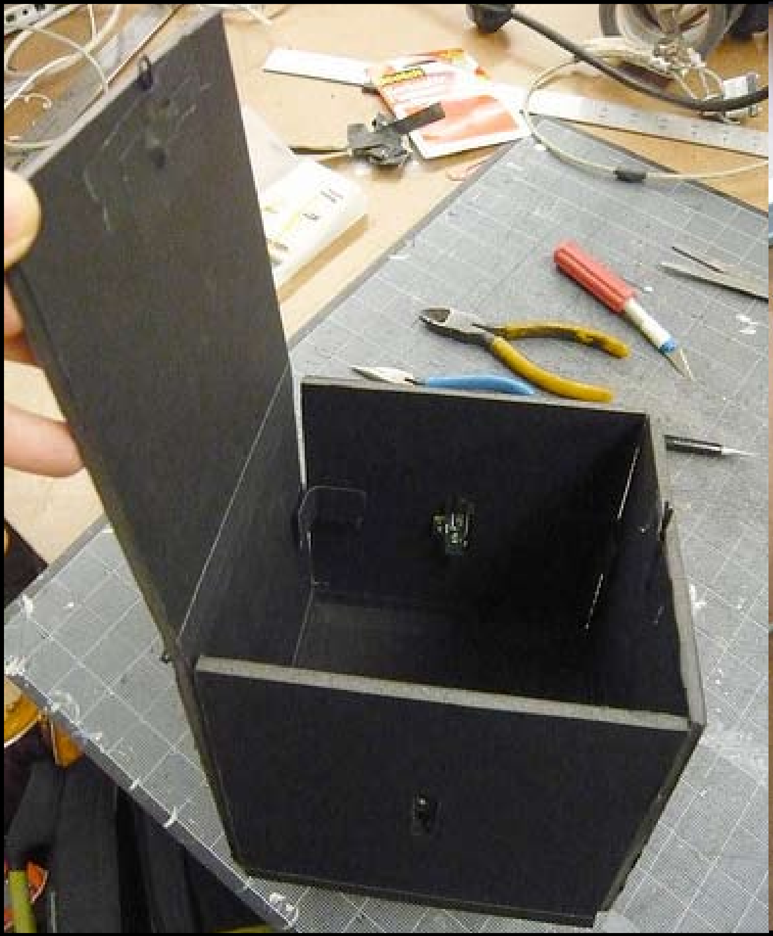
```
myPort.readStringUntil( ‘\n’
);
  //  If you got any bytes
other than the line feed:
  if ( bufferedString != null
)
  {

    if (sensors[0] > 350){
      val[0] = 1;
    } else {
      val[0] = 0;
    }
    if (sensors[1] > 350){

      val[1] = 1;
    } else {
      val[1] = 0;
    }
    if (sensors[2] > 350){
      val[2] = 1;
    } else {
      val[2] = 0;
    }
    if (sensors[3] > 350){
      val[3] = 1;
    } else {
      val[3] = 0;
    }
    if (sensors[4] > 350){
      val[4] = 1;
    } else {
      val[4] = 0;
    }
    if (sensors[5] > 350){
      val[5] = 1;
    } else {
      val[5] = 0;
```

```
//preVal=val;
//val=sensors[0];
//println(val);
    //  Loop through to read
data from each sensor.
    for ( int sensorNum = 0;
sensorNum < sensors.length;
sensorNum++ )
    {
      /*  Print out the val-
ues received from the sensors
(whose number was
          determined by the num-
ber of tabbed sections.
          */
      // print( “Sen-
sor “ + sensorNum + “: “ +
sensors[sensorNum] + “\t\t”
);
    }

}
}
void soundSample1(){
  sample1.trigger();
}

void soundSample2(){
  sample2.trigger();
}
void soundSample3(){
  sample3.trigger();
}
void soundSample4(){
  sample4.trigger();
}
void soundSample5(){
  sample5.trigger();
}
void soundSample6(){
  sample6.trigger();
}

void stop()
{
  // always close Minim audio
classes when you are done
with them
  sample1.close();
  sample2.close();
  sample3.close();
  sample4.close();
  sample5.close();
  sample6.close();
```

Design Decisions

- Each side has a different icon and color, to aid the players in associating each side with a specific sound.
- Each side uses the same kind of sensor, having the same range, to make the interaction with MC SQUARED predictable and intuitive.
- Each sensor functions as a digital switch rather than a variable control, to make it simple to operate.

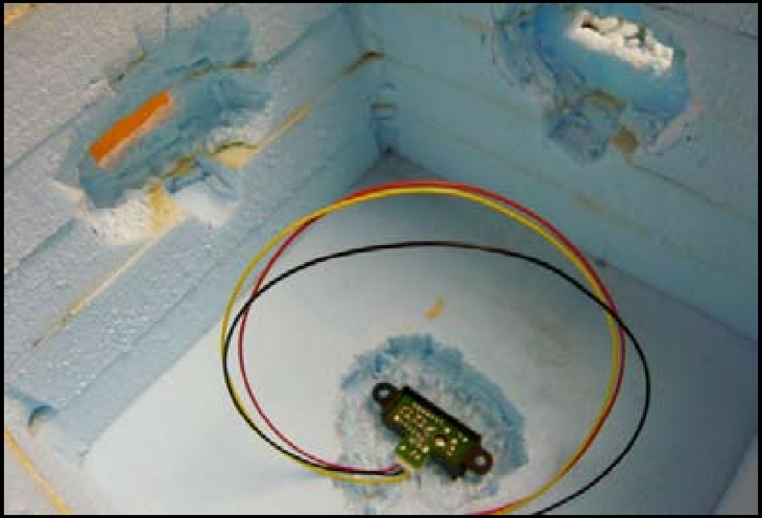
Diego demonstrating how the MC SQUARED (here represented by an iPod box) stands on its corner so that players can reach all sides.

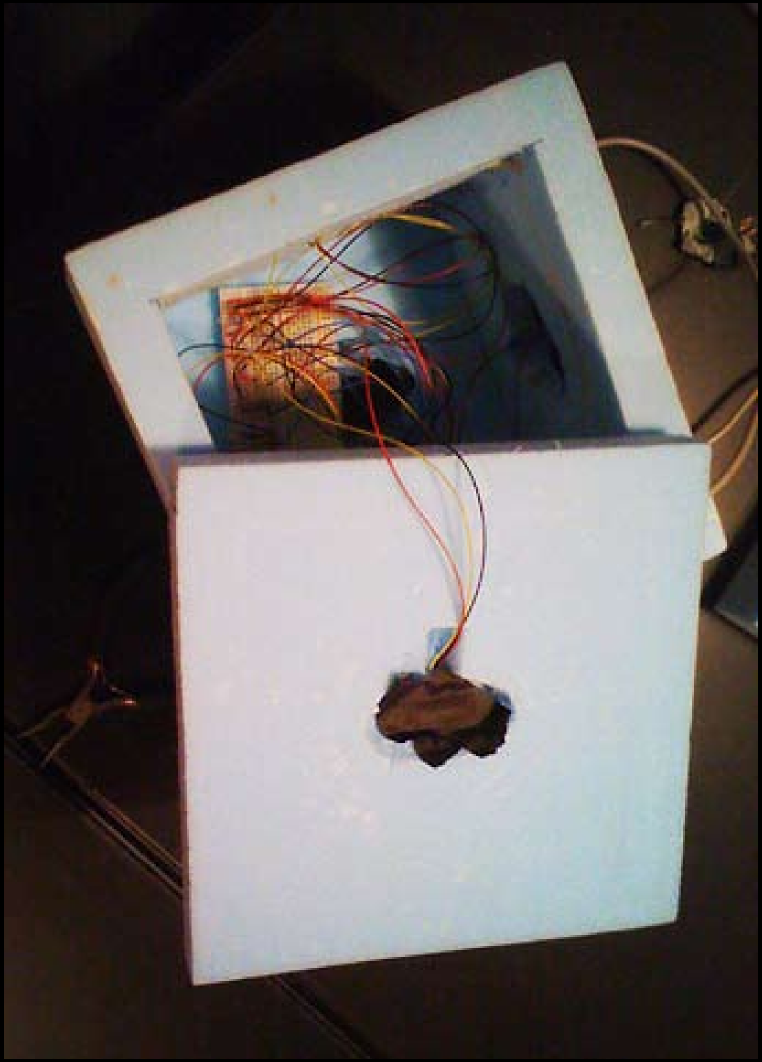


Our first box, made of black foam core, held together with black fabric tape, Velcro squares (so the whole box can be opened out flat for servicing), and a wire latch. This box measures 6 inches on each side, which turned out to be an extremely tight fit for an Arduino Diecimila and a small breadboard.

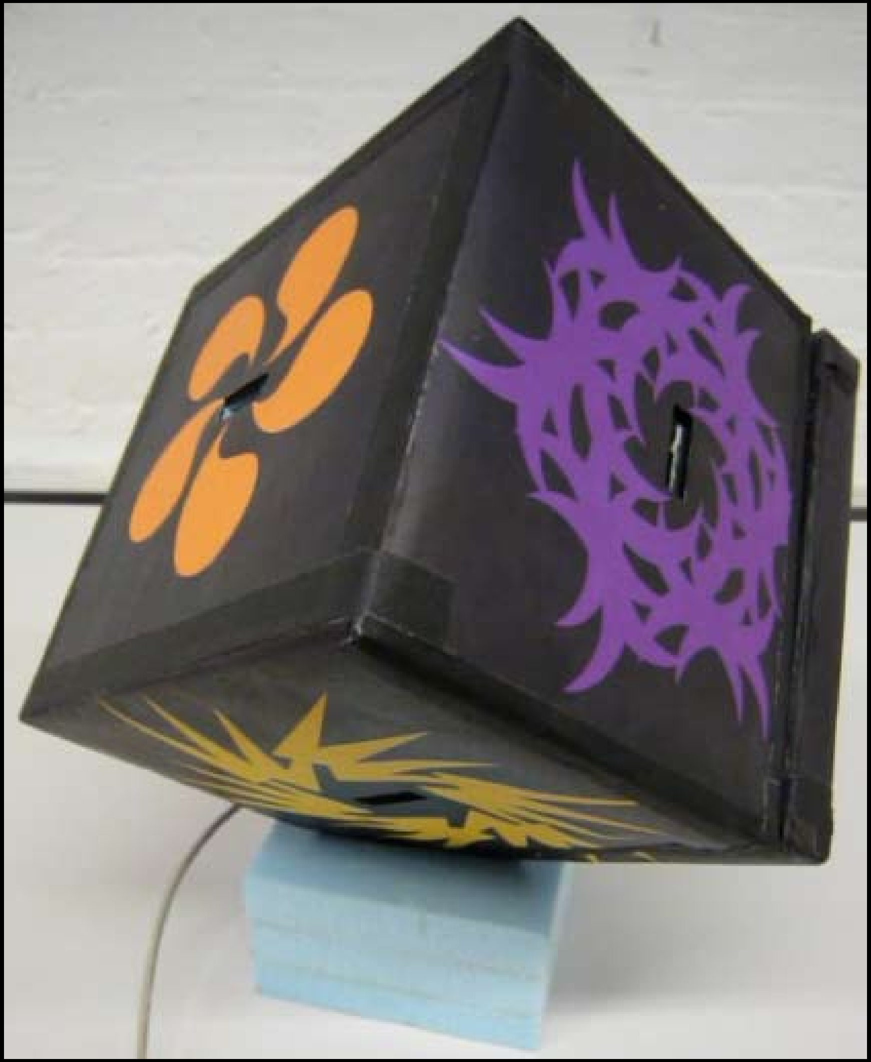
The final box was constructed from sheets of 7/8-inch urethane foam which we cut on a band saw to 8-inch squares. Six of these squares have the centers cut out. The whole stack—except for the lid—was glued together by Diego, aka The Human Clamp.

ITP



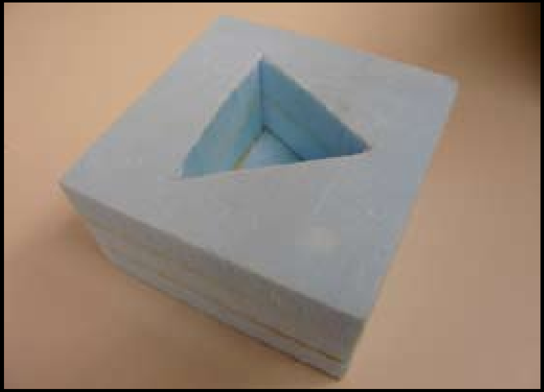


Diego learned that spraypainting the foam would cause it to break down, and we didn't have time to wait for paint to dry, anyway. So we covered the sides with printed paper, which is secured with a ton of black fabric tape. It's loose around the sensors; a dab of glue would have helped.



One view of the box we used in our presentation.

A corner of the box gets wedged into this block so that players can reach all sides. The block can be mounted on a platform (what we did) or tripod.



Future possibilities

- MC SQUARED could be installed in public places, to get people involved in playful interactions in unexpected environments (e.g., airports, train stations).
- MC SQUARED could come with alternate sound packs—electronica, stringed instruments, etc.
- Buttons and a mic could be added so that users could record new sound loops on the fly.

Thanks to all who helped play, helped code, lent us header pins, showed us how to unlock the band saw blade, and much, much more. And special thanks to Yasser Ansari, who suggested the name

mc squared