```
/*
Lee Sichello
200259098
CS 207
Project: Wii Labyrinth
Last update Nov 7, 2012

Here's what it does:
•       When booted, it will play the Mortal Kombat introduction theme from the original SNES game.
        When the theme reaches the part that says "FIGHT!" , the solenoid fires and kicks the marble
        up onto the playing field.
•       The default mode of operation for the Nunchuck is using the joystick.
        The user can press the "C" button during gameplay to switch and use the accelerometers
        for control: doing so will play also play an audio clip of a woman saying
         "Accelerometer Control" (the clip was sampled from Google Translate using Audacity).
        Pressing "C" again will switch back to the joystick and play an audio clip of a woman
       saying "Joystick Control"
•       If the ball falls through a hole, it is always successfully collected by the catch basin.
        The user can press the "Z" button to fire the solenoid and kick the ball back up again.
       However, before the solenoid is fired an audio clip (sampled from Mortal Kombat) saying
      " Round One, FIGHT!" , is played.

Pinout Notes:
  Wii Nunchuck Configuration:
      Data = pin A4
      CLK = pin A5
  Servo Wire Colors:
       Orange - Signal (pins A1,A2)
      Red - 5V
      Brown - GND
  Solenoid Signal on pin A3
*/
#include  <FatReader.h>
#include  <SdReader.h>
#include  <avr/pgmspace.h>
#include "WaveUtil.h"
#include "WaveHC.h"
#include<String.h>
//#include <Servo.h>
#include "Wire.h"
#include "WiiChuck.h"
#include "SoftwareServo.h"
#include "MsTimer2.h"

#define MAXANGLE 90
#define MINANGLE -90

const int SRV1_CENTER = 80;
const int SRV2_CENTER = 80;

WiiChuck chuck = WiiChuck();
int angleStart, currentAngle;
int tillerStart = 0;
double angle;

int  solenoid=A0;
SoftwareServo servo1;   // create servo object to control a servo
SoftwareServo servo2;  // a maximum of eight servo objects can be created
int  srv1_pos   =   90;// variable to store the servo position
int srv2_pos = 90;
int    delta1=0;// deltas represent difference between current servo angle and destination angle
int    delta2=0;// (one for each axis)

boolean  useAccel=false;
boolean  first_play=true;


SdReader card;     // This object holds the information for the card
FatVolume vol;     // This holds the information for the partition on the card
FatReader root;    // This holds the information for the filesystem on the card
FatReader f;       // This holds the information for the file we're play
```

```
WaveHC        wave;// This is the only wave (audio) object, since we will only play one at a time

// this handy function will return the number of bytes currently free in RAM, great for debugging!
int  freeRam(void)
{
   externint    __bss_end;
   externint    *__brkval;
  int   free_memory;
  if((int)__brkval == 0) {
    free_memory = ((int)&free_memory) - (( int)&__bss_end);
  }
  else {
    free_memory = ((int)&free_memory) - (( int)__brkval);
  }
  return   free_memory;
}

void  sdErrorCheck(void)
{
  if   (!card.errorCode())return;
   putstring("\n\rSD I/O error: ");
  Serial.print(card.errorCode(),  HEX);
   putstring(", ");
  Serial.println(card.errorData(),  HEX);
  while(1);
}

////////////////////////////////////////////////////////////////////////////////////
void setup() {
  byte  i;

  // set up serial port
  Serial.begin(57600);
   putstring_nl("WaveHC ");

   putstring("Free RAM: ");       // This can help with debugging, running out of RAM is bad
  Serial.println(freeRam());       // if this is under 150 bytes it may spell trouble!

  // Set the output pins for the DAC control. This pins are defined in the library
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);

  // pin13 LED
  pinMode(13, OUTPUT);

  //  if (!card.init(true)) { //play with 4 MHz spi if 8MHz isn't working for you
  if     (!card.init())       {//play with 8 MHz spi (default faster!)
     putstring_nl("Card init. failed!");  // Something went wrong, lets print out why
     sdErrorCheck();
    while(1);                              // then 'halt' - do nothing!
  }

  // enable optimize read - some cards may timeout. Disable if you're having problems
   card.partialBlockRead(true);

// Now we will look for a FAT partition!
   uint8_t part;
  for  (part  =  0;  part  <  5;  part++)  {// we have up to 5 slots to look in
    if  (vol.init(card,  part))
      break;                               // we found one, lets bail
  }
  if     (part       ==       5)       {// if we ended up not finding one  :(
     putstring_nl("No valid FAT partition!");
           sdErrorCheck();// Something went wrong, lets print out why
    while(1);                              // then 'halt' - do nothing!
  }

  // Lets tell the user about what we found
   putstring("Using partition ");
  Serial.print(part, DEC);
```

```
  putstring(", type is FAT");
  Serial.println(vol.fatType(), DEC);     // FAT16 or FAT32?

  // Try to open the root directory
  if (!root.openRoot(vol)) {
    putstring_nl("Can't open root dir!"); // Something went wrong,
    while(1);                             // then 'halt' - do nothing!
  }

  // Whew! We got past the tough parts.
  putstring_nl("Ready!");

  chuck.begin();
  chuck.update();
  //chuck.calibrateJoy();

  //establish solenoid as output
  pinMode(solenoid, OUTPUT);
  analogWrite(solenoid,LOW);

  servo1.attach(A1);  // attaches the servo on pin A1 to the servo object
  servo2.attach(A2);  // attaches the servo on pin A2 to the servo object

  MsTimer2::set(10,   SoftwareServo::refresh);// 500ms period
  MsTimer2::start();

  servo1.write(srv1_pos);  //update servo position
  servo2.write(srv2_pos);
}

//================================================================================================================
void loop() {
   chuck.update();
  delay(40);

  //play mortal kombat theme on the first play only, then fire solenoid to start
  if(first_play){
    first_play=false;
    playcomplete("mortal.wav");
    delay(2);
    digitalWrite(solenoid, HIGH);
    delay(15);
    digitalWrite(solenoid, LOW);
    delay(500);
  }

  //check if the C button on Nunchuck was pressed, if so, toggle between accelerometers and joystick
  if(chuck.buttonC){
   if (useAccel==true){
     useAccel=false;
     playcomplete("joystick.wav");
   }
   else{
    useAccel=true;
    playcomplete("accel.wav");
   }
  }

  //check if the Z button on Nunchuck was pressed, if so, fire the solenoid
  if(chuck.buttonZ){
    playcomplete("fight.wav");
    delay(2);
    digitalWrite(solenoid, HIGH);
    delay(15);
    digitalWrite(solenoid, LOW);
    delay(500);
  }


  //if using the accelerometers find the diference between current accelrometer X angle and servo position
  if(useAccel){
    //divide by 5 to slowly approach the destination position
```

```
      delta1 = (chuck.readAccelX()*0.9 + SRV1_CENTER -srv1_pos)/5;
      delta2 = (chuck.readAccelY()*0.9 + SRV2_CENTER -srv2_pos)/5;
   }
    //else use the joystick to find the diference between current joystick X angle and servo position
   else{
     //divide by 3 to slowly approach the destination position
      delta1 = (chuck.readJoyX()*0.9 + SRV1_CENTER -srv1_pos)/3;
      delta2 = (chuck.readJoyY()*0.9 + SRV2_CENTER -srv2_pos)/3;
   }

    srv1_pos =srv1_pos + delta1;//add difference to correct position
   srv2_pos =srv2_pos + delta2;
   servo1.write(srv1_pos);   //update servo position
   servo2.write(srv2_pos);


}

//================================================================================================================

// Plays a full file from beginning to end with no pause.
void playcomplete(char *name) {
  // call our helper to find and play this name
   playfile(name);
  while (wave.isplaying) {
  // do nothing while its playing
  }
  // now its done playing
}

void playfile(char *name) {
  // see if the wave object is currently doing something
  if (wave.isplaying) {// already playing something, so stop it!
    wave.stop(); // stop it
  }
  // look in the root directory and open the file
  if (!f.open(root, name)) {
    putstring("Couldn't open file "); Serial.print(name); return;
  }
  // OK read the file and turn it into a wave object
  if (!wave.create(f)) {
    putstring_nl("Not a valid WAV"); return;
  }

  // ok time to play! start playback
   wave.play();
}
```