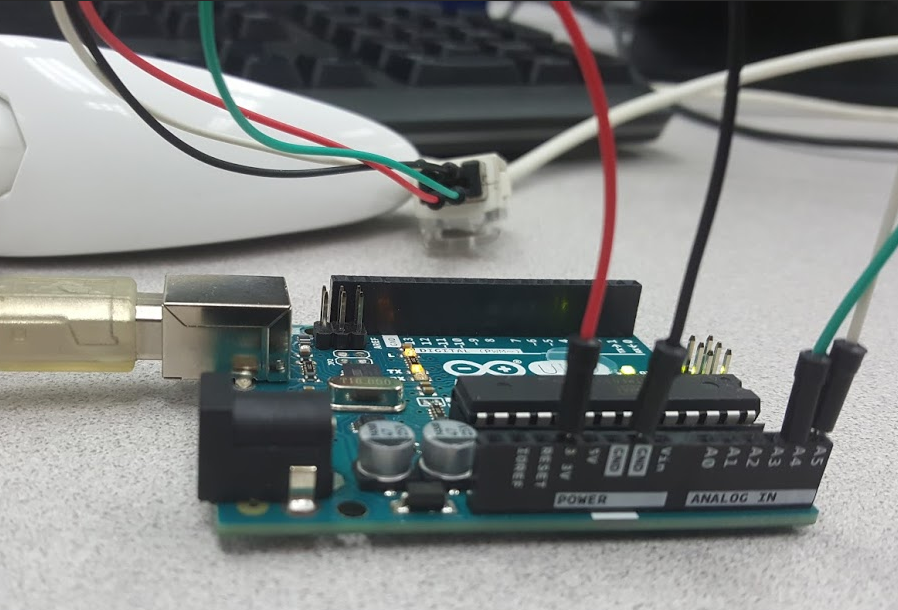
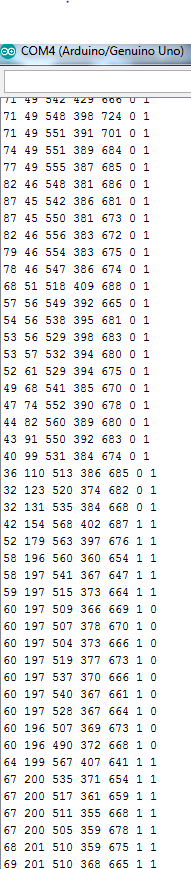
Sarah Wood

Embedded Programming

Chapter 9

Assignment 1

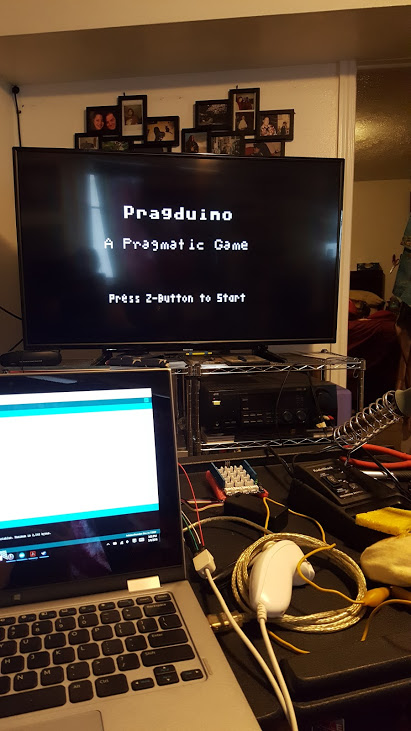
Nintendo Wii Nunchuk controller



Basic Hardware functionality was pretty simple.

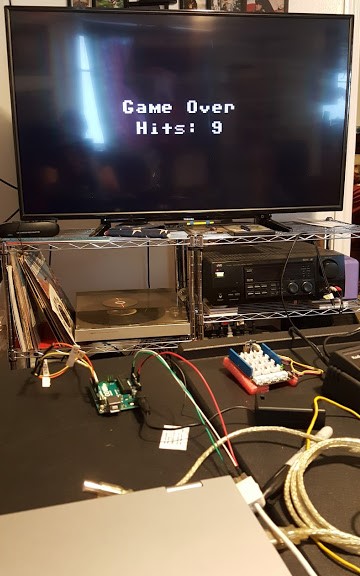
(Am I the only one who gets a little nervous if a new thing works the first time?)

Here is a pic of the Pragduino code working on my TV.



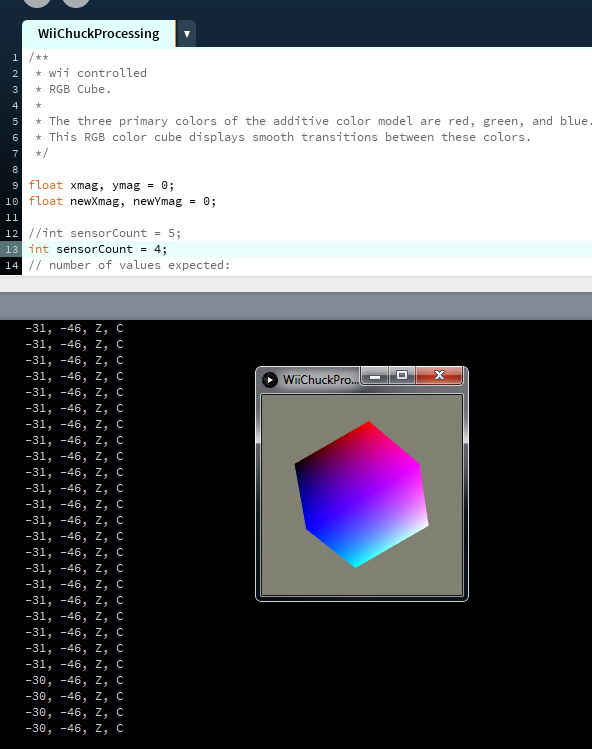


This is hard. The tracking is smooth, but slow. What if I give myself more time?



OK, maybe that’s too easy. I’ll fine-tune the game controls at some point.

(Bonus!) I got the 3D cube code from Arduino Playground. It worked great, and was very responsive.



The source code for basic functionality is straight out of the book. Here it is for reference.

Main.cpp

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <Wire.h>

#include "nunchuk.h"

const unsigned int BAUD\_RATE = 19200;

Nunchuk nunchuk;

void setup() {

Serial.begin(BAUD\_RATE);

nunchuk.initialize();

}

void loop() {

if (nunchuk.update()) {

Serial.print(nunchuk.joystick\_x());

Serial.print(" ");

Serial.print(nunchuk.joystick\_y());

Serial.print(" ");

Serial.print(nunchuk.x\_acceleration());

Serial.print(" ");

Serial.print(nunchuk.y\_acceleration());

Serial.print(" ");

Serial.print(nunchuk.z\_acceleration());

Serial.print(" ");

Serial.print(nunchuk.z\_button());

Serial.print(" ");

Serial.println(nunchuk.c\_button());

}

}

Nunchuck.cpp

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <Arduino.h>

#include <Wire.h>

#include "nunchuk.h"

#define NUNCHUK\_DEVICE\_ID 0x52

void Nunchuk::initialize() {

Wire.begin();

Wire.beginTransmission(NUNCHUK\_DEVICE\_ID);

Wire.write((byte)0x40);

Wire.write((byte)0x00);

Wire.endTransmission();

update();

}

bool Nunchuk::update() {

delay(1);

Wire.requestFrom(NUNCHUK\_DEVICE\_ID, NUNCHUK\_BUFFER\_SIZE);

int byte\_counter = 0;

while (Wire.available() && byte\_counter < NUNCHUK\_BUFFER\_SIZE)

\_buffer[byte\_counter++] = decode\_byte(Wire.read());

request\_data();

return byte\_counter == NUNCHUK\_BUFFER\_SIZE;

}

void Nunchuk::request\_data() {

Wire.beginTransmission(NUNCHUK\_DEVICE\_ID);

Wire.write((byte)0x00);

Wire.endTransmission();

}

char Nunchuk::decode\_byte(const char b) {

return (b ^ 0x17) + 0x17;

}

Nunchuck.h

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/\*\*\*

\* Excerpted from "Arduino: A Quick-Start Guide, Second Edition",

\* published by The Pragmatic Bookshelf.

\* Copyrights apply to this code. It may not be used to create training material,

\* courses, books, articles, and the like. Contact us if you are in doubt.

\* We make no guarantees that this code is fit for any purpose.

\* Visit http://www.pragmaticprogrammer.com/titles/msard2 for more book information.

\*\*\*/

#ifndef \_\_NUNCHUK\_H\_\_

#define \_\_NUNCHUK\_H\_\_

#define NUNCHUK\_BUFFER\_SIZE 6

class Nunchuk {

public:

void initialize();

bool update();

int joystick\_x() const { return \_buffer[0]; }

int joystick\_y() const { return \_buffer[1]; }

int x\_acceleration() const {

return ((int)(\_buffer[2]) << 2) | ((\_buffer[5] >> 2) & 0x03);

}

int y\_acceleration() const {

return ((int)(\_buffer[3]) << 2) | ((\_buffer[5] >> 4) & 0x03);

}

int z\_acceleration() const {

return ((int)(\_buffer[4]) << 2) | ((\_buffer[5] >> 6) & 0x03);

}

bool z\_button() const { return !(\_buffer[5] & 0x01); }

bool c\_button() const { return !(\_buffer[5] & 0x02); }

private:

void request\_data();

char decode\_byte(const char);

unsigned char \_buffer[NUNCHUK\_BUFFER\_SIZE];

};

#endif

Pragduino is also straight out of the book. Here it is for reference.

Pragduino uses the “Nunchuck.cpp” and “Nunchuck.h” files as included above.

Pragduino.cpp

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

#include <Wire.h>

#include <TVout.h>

#include <fontALL.h>

#include "nunchuk.h"

const int WIDTH = 128;

const int HEIGHT = 96;

const int CH\_LEN = 8;

const int MAX\_TARGET = 10;

const int TARGET\_LIFESPAN = 1500;

TVout tv;

Nunchuk nunchuk;

boolean up, down, left, right, c\_button, z\_button;

int chx, chy;

int chvx, chvy;

int target\_x, target\_y, target\_r;

unsigned int target\_count;

unsigned int hits;

unsigned long target\_creation;

enum GameState {

INTRO, STARTING, RUNNING, DONE

};

GameState state;

void init\_game() {

up = down = left = right = c\_button = z\_button = false;

chx = WIDTH / 2;

chy = HEIGHT / 2;

chvx = 1;

chvy = 1;

state = INTRO;

target\_count = 0;

hits = 0;

create\_target();

}

void create\_target() {

target\_r = random(7, 11);

target\_x = random(target\_r, WIDTH - target\_r);

target\_y = random(target\_r, HEIGHT - target\_r);

target\_count++;

target\_creation = millis();

}

void setup() {

randomSeed(analogRead(A0));

tv.begin(PAL, WIDTH, HEIGHT);

nunchuk.initialize();

init\_game();

}

void loop() {

check\_controls();

switch (state) {

case INTRO: intro(); break;

case STARTING: start\_game(); break;

case RUNNING: update\_game(); break;

case DONE: game\_over(); break;

}

tv.delay\_frame(1);

}

void check\_controls() {

up = down = left = right = c\_button = z\_button = false;

if (nunchuk.update())

{

if (nunchuk.joystick\_x() < 70)

left = true;

if (nunchuk.joystick\_x() > 150)

right = true;

if (nunchuk.joystick\_y() > 150)

up = true;

if (nunchuk.joystick\_y() < 70)

down = true;

c\_button = nunchuk.c\_button();

z\_button = nunchuk.z\_button();

}

}

void move\_crosshairs() {

if (left) chx -= chvx;

if (right) chx += chvx;

if (up) chy -= chvy;

if (down) chy += chvy;

if (chx <= CH\_LEN)

chx = CH\_LEN + 1;

if (chx >= WIDTH - CH\_LEN)

chx = WIDTH - CH\_LEN - 1;

if (chy <= CH\_LEN)

chy = CH\_LEN + 1;

if (chy >= HEIGHT - CH\_LEN)

chy = HEIGHT - CH\_LEN - 1;

}

void draw\_crosshairs() {

tv.draw\_row(chy, chx - CH\_LEN, chx - 1, WHITE);

tv.draw\_row(chy, chx + 1, chx + CH\_LEN, WHITE);

tv.draw\_column(chx, chy - CH\_LEN, chy - 1, WHITE);

tv.draw\_column(chx, chy + 1, chy + CH\_LEN, WHITE);

}

bool target\_hit() {

if (z\_button)

return (target\_x - chx) \* (target\_x - chx) +

(target\_y - chy) \* (target\_y - chy) < target\_r \* target\_r;

return false;

}

void check\_target() {

if (target\_hit()) {

hits++;

create\_target();

}

int remaining\_time = millis() - target\_creation;

if (remaining\_time >= TARGET\_LIFESPAN) {

create\_target();

}

int w = map(TARGET\_LIFESPAN - remaining\_time, 0, TARGET\_LIFESPAN, 0, WIDTH);

tv.draw\_rect(0, 0, w, 3, WHITE, WHITE);

}

void intro() {

tv.select\_font(font8x8);

tv.printPGM(28, 20, PSTR("Pragduino"));

tv.select\_font(font6x8);

tv.printPGM(16, 40, PSTR("A Pragmatic Game"));

tv.select\_font(font4x6);

tv.printPGM(18, 74, PSTR("Press Z-Button to Start"));

if (z\_button) {

state = STARTING;

z\_button = false;

delay(200);

}

}

void start\_game() {

tv.clear\_screen();

tv.select\_font(font8x8);

tv.printPGM(40, 44, PSTR("READY?"));

if (z\_button) {

init\_game();

state = RUNNING;

}

}

void game\_over() {

tv.clear\_screen();

tv.select\_font(font8x8);

tv.printPGM(28, 38, PSTR("Game Over"));

int x = (WIDTH - 7 \* 8) / 2;

if (hits > 9)

x = (WIDTH - 8 \* 8) / 2;

tv.printPGM(x, 50, PSTR("Hits: "));

tv.print(x + 6 \* 8, 50, hits);

if (z\_button) {

state = STARTING;

z\_button = false;

delay(200);

}

}

void update\_game() {

tv.clear\_screen();

tv.draw\_circle(target\_x, target\_y, target\_r, WHITE);

move\_crosshairs();

draw\_crosshairs();

check\_target();

if (target\_count == MAX\_TARGET + 1) {

state = DONE;

z\_button = false;

delay(200);

}

}