#include <LiquidCrystal.h>

#define PIN\_BUTTON 2

#define PIN\_AUTOPLAY 1

#define PIN\_READWRITE 10

#define PIN\_CONTRAST 12

#define SPRITE\_RUN1 1

#define SPRITE\_RUN2 2

#define SPRITE\_JUMP 3

#define SPRITE\_JUMP\_UPPER '.' // Use the '.' character for the head

#define SPRITE\_JUMP\_LOWER 4

#define SPRITE\_TERRAIN\_EMPTY ' ' // User the ' ' character

#define SPRITE\_TERRAIN\_SOLID 5

#define SPRITE\_TERRAIN\_SOLID\_RIGHT 6

#define SPRITE\_TERRAIN\_SOLID\_LEFT 7

#define HERO\_HORIZONTAL\_POSITION 1 // Horizontal position of hero on screen

#define TERRAIN\_WIDTH 16

#define TERRAIN\_EMPTY 0

#define TERRAIN\_LOWER\_BLOCK 1

#define TERRAIN\_UPPER\_BLOCK 2

#define HERO\_POSITION\_OFF 0 // Hero is invisible

#define HERO\_POSITION\_RUN\_LOWER\_1 1 // Hero is running on lower row (pose 1)

#define HERO\_POSITION\_RUN\_LOWER\_2 2 // (pose 2)

#define HERO\_POSITION\_JUMP\_1 3 // Starting a jump

#define HERO\_POSITION\_JUMP\_2 4 // Half-way up

#define HERO\_POSITION\_JUMP\_3 5 // Jump is on upper row

#define HERO\_POSITION\_JUMP\_4 6 // Jump is on upper row

#define HERO\_POSITION\_JUMP\_5 7 // Jump is on upper row

#define HERO\_POSITION\_JUMP\_6 8 // Jump is on upper row

#define HERO\_POSITION\_JUMP\_7 9 // Half-way down

#define HERO\_POSITION\_JUMP\_8 10 // About to land

#define HERO\_POSITION\_RUN\_UPPER\_1 11 // Hero is running on upper row (pose 1)

#define HERO\_POSITION\_RUN\_UPPER\_2 12 // (pose 2)

LiquidCrystal lcd(11, 9, 6, 5, 4, 3);

static char terrainUpper[TERRAIN\_WIDTH + 1];

static char terrainLower[TERRAIN\_WIDTH + 1];

static bool buttonPushed = false;

void initializeGraphics(){

static byte graphics[] = {

// Run position 1

B01100,

B01100,

B00000,

B01110,

B11100,

B01100,

B11010,

B10011,

// Run position 2

B01100,

B01100,

B00000,

B01100,

B01100,

B01100,

B01100,

B01110,

// Jump

B01100,

B01100,

B00000,

B11110,

B01101,

B11111,

B10000,

B00000,

// Jump lower

B11110,

B01101,

B11111,

B10000,

B00000,

B00000,

B00000,

B00000,

// Ground

B11111,

B11111,

B11111,

B11111,

B11111,

B11111,

B11111,

B11111,

// Ground right

B00011,

B00011,

B00011,

B00011,

B00011,

B00011,

B00011,

B00011,

// Ground left

B11000,

B11000,

B11000,

B11000,

B11000,

B11000,

B11000,

B11000,

};

int i;

// Skip using character 0, this allows lcd.print() to be used to

// quickly draw multiple characters

for (i = 0; i < 7; ++i) {

lcd.createChar(i + 1, &graphics[i \* 8]);

}

for (i = 0; i < TERRAIN\_WIDTH; ++i) {

terrainUpper[i] = SPRITE\_TERRAIN\_EMPTY;

terrainLower[i] = SPRITE\_TERRAIN\_EMPTY;

}

}

// Slide the terrain to the left in half-character increments

//

void advanceTerrain(char\* terrain, byte newTerrain){

for (int i = 0; i < TERRAIN\_WIDTH; ++i) {

char current = terrain[i];

char next = (i == TERRAIN\_WIDTH-1) ? newTerrain : terrain[i+1];

switch (current){

case SPRITE\_TERRAIN\_EMPTY:

terrain[i] = (next == SPRITE\_TERRAIN\_SOLID) ? SPRITE\_TERRAIN\_SOLID\_RIGHT : SPRITE\_TERRAIN\_EMPTY;

break;

case SPRITE\_TERRAIN\_SOLID:

terrain[i] = (next == SPRITE\_TERRAIN\_EMPTY) ? SPRITE\_TERRAIN\_SOLID\_LEFT : SPRITE\_TERRAIN\_SOLID;

break;

case SPRITE\_TERRAIN\_SOLID\_RIGHT:

terrain[i] = SPRITE\_TERRAIN\_SOLID;

break;

case SPRITE\_TERRAIN\_SOLID\_LEFT:

terrain[i] = SPRITE\_TERRAIN\_EMPTY;

break;

}

}

}

bool drawHero(byte position, char\* terrainUpper, char\* terrainLower, unsigned int score) {

bool collide = false;

char upperSave = terrainUpper[HERO\_HORIZONTAL\_POSITION];

char lowerSave = terrainLower[HERO\_HORIZONTAL\_POSITION];

byte upper, lower;

switch (position) {

case HERO\_POSITION\_OFF:

upper = lower = SPRITE\_TERRAIN\_EMPTY;

break;

case HERO\_POSITION\_RUN\_LOWER\_1:

upper = SPRITE\_TERRAIN\_EMPTY;

lower = SPRITE\_RUN1;

break;

case HERO\_POSITION\_RUN\_LOWER\_2:

upper = SPRITE\_TERRAIN\_EMPTY;

lower = SPRITE\_RUN2;

break;

case HERO\_POSITION\_JUMP\_1:

case HERO\_POSITION\_JUMP\_8:

upper = SPRITE\_TERRAIN\_EMPTY;

lower = SPRITE\_JUMP;

break;

case HERO\_POSITION\_JUMP\_2:

case HERO\_POSITION\_JUMP\_7:

upper = SPRITE\_JUMP\_UPPER;

lower = SPRITE\_JUMP\_LOWER;

break;

case HERO\_POSITION\_JUMP\_3:

case HERO\_POSITION\_JUMP\_4:

case HERO\_POSITION\_JUMP\_5:

case HERO\_POSITION\_JUMP\_6:

upper = SPRITE\_JUMP;

lower = SPRITE\_TERRAIN\_EMPTY;

break;

case HERO\_POSITION\_RUN\_UPPER\_1:

upper = SPRITE\_RUN1;

lower = SPRITE\_TERRAIN\_EMPTY;

break;

case HERO\_POSITION\_RUN\_UPPER\_2:

upper = SPRITE\_RUN2;

lower = SPRITE\_TERRAIN\_EMPTY;

break;

}

if (upper != ' ') {

terrainUpper[HERO\_HORIZONTAL\_POSITION] = upper;

collide = (upperSave == SPRITE\_TERRAIN\_EMPTY) ? false : true;

}

if (lower != ' ') {

terrainLower[HERO\_HORIZONTAL\_POSITION] = lower;

collide |= (lowerSave == SPRITE\_TERRAIN\_EMPTY) ? false : true;

}

byte digits = (score > 9999) ? 5 : (score > 999) ? 4 : (score > 99) ? 3 : (score > 9) ? 2 : 1;

// Draw the scene

terrainUpper[TERRAIN\_WIDTH] = '\0';

terrainLower[TERRAIN\_WIDTH] = '\0';

char temp = terrainUpper[16-digits];

terrainUpper[16-digits] = '\0';

lcd.setCursor(0,0);

lcd.print(terrainUpper);

terrainUpper[16-digits] = temp;

lcd.setCursor(0,1);

lcd.print(terrainLower);

lcd.setCursor(16 - digits,0);

lcd.print(score);

terrainUpper[HERO\_HORIZONTAL\_POSITION] = upperSave;

terrainLower[HERO\_HORIZONTAL\_POSITION] = lowerSave;

return collide;

}

// Handle the button push as an interrupt

void buttonPush() {

buttonPushed = true;

}

void setup(){

pinMode(PIN\_READWRITE, OUTPUT);

digitalWrite(PIN\_READWRITE, LOW);

pinMode(PIN\_CONTRAST, OUTPUT);

digitalWrite(PIN\_CONTRAST, LOW);

pinMode(PIN\_BUTTON, INPUT);

digitalWrite(PIN\_BUTTON, HIGH);

pinMode(PIN\_AUTOPLAY, OUTPUT);

digitalWrite(PIN\_AUTOPLAY, HIGH);

// Digital pin 2 maps to interrupt 0

attachInterrupt(0/\*PIN\_BUTTON\*/, buttonPush, FALLING);

initializeGraphics();

lcd.begin(16, 2);

}

void loop(){

static byte heroPos = HERO\_POSITION\_RUN\_LOWER\_1;

static byte newTerrainType = TERRAIN\_EMPTY;

static byte newTerrainDuration = 1;

static bool playing = false;

static bool blink = false;

static unsigned int distance = 0;

if (!playing) {

drawHero((blink) ? HERO\_POSITION\_OFF : heroPos, terrainUpper, terrainLower, distance >> 3);

if (blink) {

lcd.setCursor(0,0);

lcd.print("Game Over");

}

delay(250);

blink = !blink;

if (buttonPushed) {

initializeGraphics();

heroPos = HERO\_POSITION\_RUN\_LOWER\_1;

playing = true;

buttonPushed = false;

distance = 0;

}

return;

}

// Shift the terrain to the left

advanceTerrain(terrainLower, newTerrainType == TERRAIN\_LOWER\_BLOCK ? SPRITE\_TERRAIN\_SOLID : SPRITE\_TERRAIN\_EMPTY);

advanceTerrain(terrainUpper, newTerrainType == TERRAIN\_UPPER\_BLOCK ? SPRITE\_TERRAIN\_SOLID : SPRITE\_TERRAIN\_EMPTY);

// Make new terrain to enter on the right

if (--newTerrainDuration == 0) {

if (newTerrainType == TERRAIN\_EMPTY) {

newTerrainType = (random(3) == 0) ? TERRAIN\_UPPER\_BLOCK : TERRAIN\_LOWER\_BLOCK;

newTerrainDuration = 2 + random(10);

} else {

newTerrainType = TERRAIN\_EMPTY;

newTerrainDuration = 10 + random(10);

}

}

if (buttonPushed) {

if (heroPos <= HERO\_POSITION\_RUN\_LOWER\_2) heroPos = HERO\_POSITION\_JUMP\_1;

buttonPushed = false;

}

if (drawHero(heroPos, terrainUpper, terrainLower, distance >> 3)) {

playing = false; // The hero collided with something. Too bad.

} else {

if (heroPos == HERO\_POSITION\_RUN\_LOWER\_2 || heroPos == HERO\_POSITION\_JUMP\_8) {

heroPos = HERO\_POSITION\_RUN\_LOWER\_1;

} else if ((heroPos >= HERO\_POSITION\_JUMP\_3 && heroPos <= HERO\_POSITION\_JUMP\_5) && terrainLower[HERO\_HORIZONTAL\_POSITION] != SPRITE\_TERRAIN\_EMPTY) {

heroPos = HERO\_POSITION\_RUN\_UPPER\_1;

} else if (heroPos >= HERO\_POSITION\_RUN\_UPPER\_1 && terrainLower[HERO\_HORIZONTAL\_POSITION] == SPRITE\_TERRAIN\_EMPTY) {

heroPos = HERO\_POSITION\_JUMP\_5;

} else if (heroPos == HERO\_POSITION\_RUN\_UPPER\_2) {

heroPos = HERO\_POSITION\_RUN\_UPPER\_1;

} else {

++heroPos;

}

++distance;

digitalWrite(PIN\_AUTOPLAY, terrainLower[HERO\_HORIZONTAL\_POSITION + 2] == SPRITE\_TERRAIN\_EMPTY ? HIGH : LOW);

}

delay(100);

}