Bounce - an accelerometer game in Arduino for PyGamer and PyBadge

Created by Anne Barela

https://learn.adafruit.com/bounce-an-accelerometer-game-in-arduino-for-pygamer-and-pybadge

Last updated on 2024-06-03 02:48:50 PM EDT

Table of Contents

Overview

Back in ye olde 2013, Arduino came out with the Arduino Esplora. It was much like an Adafruit Circuit Playground Express and an Adafruit PyGamer rolled into one board. With an add on display, you had about a $100 self-contained handheld device (without a provision for a battery). Few games were made for the Esplora at the time. The author ported two games, Bounce and Pong, and even added an Adafruit MintyBoost to get it portable on a LiPo battery. The cost and the features did not find a market and the Esplora was quietly discontinued.

Fast forward to today: boards with modern microcontrollers and displays are proliferating, thanks to the lowering of costs and environments like MakeCode Arcade, allowing anyone to make graphical games! Several companies are making boards in this format including a number from Adafruit's PyGamer and PyBadge.

I told Ladyada I'd port one of the Arduino games from Esplora to PyGamer and she said "Go for it!". It was not difficult: if your old program used the Adafruit GFX Arduino library (and most Arduino display-based code uses that open source code), you can easily port the game to the modern Adafruit Arcada game libraries which are a superset of Adafruit GFX.

This game was originally written by R0d0t and featured in Hackaday back in 2012. It's been ported to Arcada and the scoring and controls have been improved.

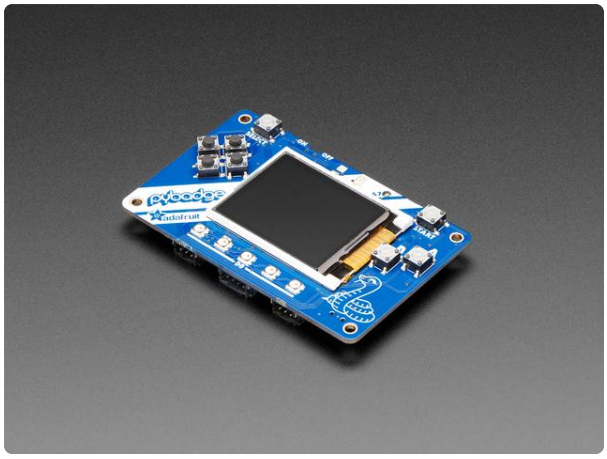
Precompiled versions are available in UF2 format to drop onto your compatible boards.

Parts

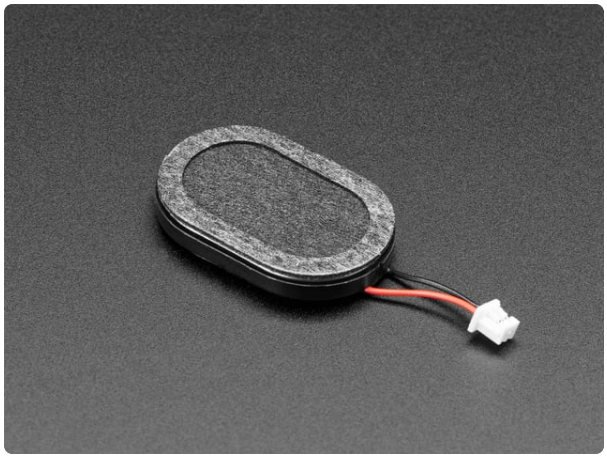
The original Adafruit PyBadge or PyGamer boards work - an accelerometer is needed so the PyBadge LC is not compatible.

Adafruit PyGamer for MakeCode Arcade, CircuitPython or Arduino
What fits in your pocket, is fully Open Source, and can run CircuitPython, MakeCode Arcade or Arduino games you write yourself? That's right, it's the Adafruit...
https://www.adafruit.com/product/4242

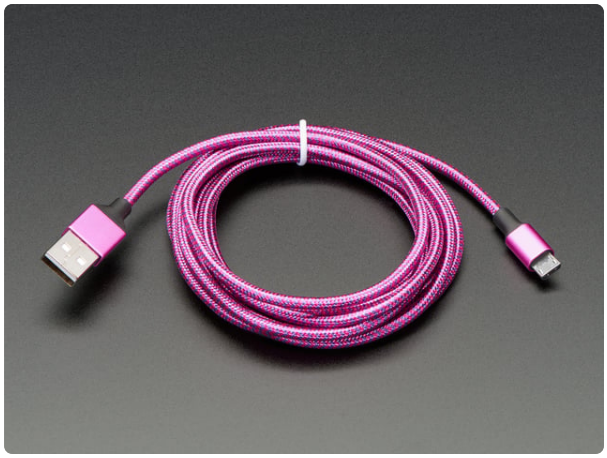
[Adafruit PyGamer Starter Kit]
Please note: you may get a royal blue or purple case with your starter kit (they're both lovely colors)What fits in your pocket, is fully Open...
https://www.adafruit.com/product/4277



[Adafruit PyBadge for MakeCode Arcade, CircuitPython, or Arduino]
What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino? That's right, its the Adafruit PyBadge! We wanted to see how much we...
https://www.adafruit.com/product/4200



[Mini Oval Speaker with Short Wires - 8 Ohm 1 Watt]
Hear the good news! This wee speaker is a great addition to any audio project where you need 8 ohm impedance and 1W or less of power. We particularly like...
https://www.adafruit.com/product/4227



[Pink and Purple Braided USB A to Micro B Cable - 2 meter long]
This cable is super-fashionable with a woven pink and purple Blinka-like pattern! First let's talk about the cover and over-molding. We got these in custom colors,...
https://www.adafruit.com/product/4148

Programming

Want to play now?

Download one of the following UF2 files for your particular device:

PyBadge_Bounce.UF2

https://adafru.it/F95

PyGamer_Bounce.UF2

https://adafru.it/F96

Double click the reset button to get to the UF2 screen. Go to the appropriate flash drive that appears on your computer - **PYBADGEBOOT** or **PYGAMERBOOT** and copy over the correct UF2 file from above and the device should reboot and start the game.

Compiling the Game in Arduino

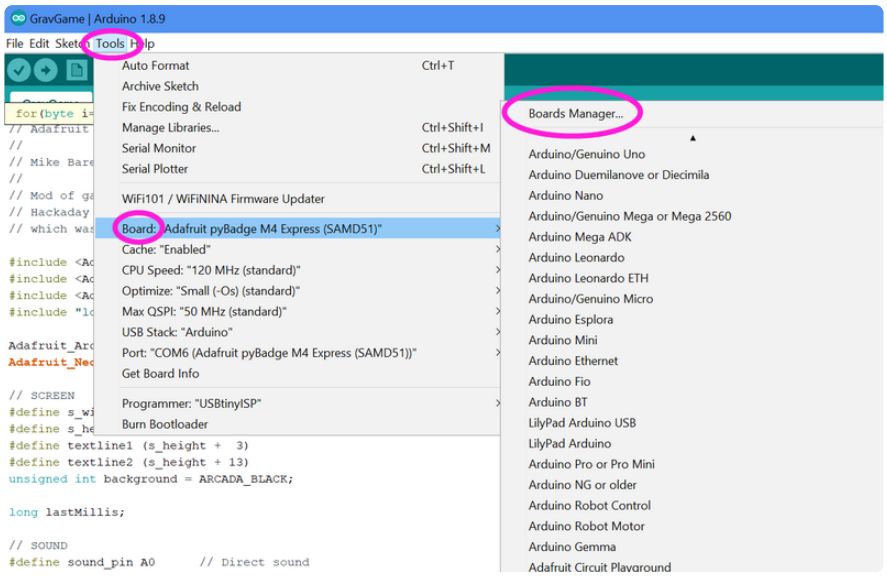
The code uses the Adafruit Arcada library. Arcada is a full featured set of game creation and display functions. It can be thought of the older Adafruit GFX library on super steroids.

In the Arduino IDE, you will need to use the Library manager to load a BUNCH of helper libraries. Please take the time to get the latest versions to ensure you have the latest code tweaks.

Guide to loading all the helper libraries required for Arcada

https://adafru.it/EUk

The code also requires a recent Adafruit SAMD boards file. Version 1.5 or higher. Go to the boards manager via the steps listed below.



Type "samd" into the search. The first couple of entries will probably not be the Adafruit package. You want to install the package called **Adafruit SAMD Boards** by **Adafruit** and be sure the version is 1.5 or higher.

This will give you the support for the Adafruit PyGamer and PyBadge definitions for the game.

Once you have the latest board definitions, you can select the **Adafruit PyGamer** or the **Adafruit PyBadge** for the game depending on your board. The PyBadge LC will not work as it does not have an accelerometer to provide the game interactivity.

Code

If you click the **Download: Zip** link, you should get both the **bounce.ino** and **logo.h** files. Save them in your Arduino projects directory in a subdirectory named **bounce**.

```
// SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// Adafruit Arcada based Level Bounce Game
//
// Anne Barela, June 3, 2019 for Adafruit Industries
//
// Mod of game by R0D0T posted on http://r0d0t.tumblr.com/post/29641975900 and
// Hackaday http://hackaday.com/2012/10/01/fantastic-programming-makes-this-arduino-
gaming-device-something-special/
// which was published for Arduino Esplora by Anne Barela 2013
```

```
#include <Adafruit_Arcada.h>
#include <Adafruit_Arcada_Def.h>
#include <Adafruit_NeoPixel.h>
#include "logo.h"

Adafruit_Arcada arcada; // Set up the Arcada code
Adafruit_NeoPixel strip(5, 8, NEO_GRB + NEO_KHZ800);  // 5 NeoPixels are on D8

// SCREEN
#define s_width   128
#define s_height  139
#define textline1 (s_height +  3)
#define textline2 (s_height + 13)
unsigned int background = ARCADA_BLACK;

long lastMillis;

// SOUND
#define sound_pin A0      // Direct sound
uint8_t sound_on = 1;     // start with sound on
char speaker_icon = 0xEC; // icon to show sound off

// PLAYER
float p_X = s_width/2;
float p_Y = s_height/2;
float p_lastX, p_vX;
float p_lastY, p_vY;
//#define p_vmaxY 6
//#define p_vmaxX 6
#define gravity_default 0.2
float gravity = 0.2;
#define p_bounceSpeed_default 4
byte p_bounceSpeed = p_bounceSpeed_default;
#define p_width_default 4
byte p_width = p_width_default;
#define p_height_default 4
byte p_height = p_height_default;
#define p_color_default ARCADA_WHITE
#define p_color_dead    ARCADA_RED
int p_color = p_color_default;

// SCORE & BONUS
unsigned int score = 0;
unsigned int highscore = 0;
byte nextBonus;
boolean b_pendingPlatform = false;
boolean b_used;
byte b_platformID;
byte b_pendingID;
byte b_onUseID;
byte b_remaining = 0;
byte b_max;
#define b_number 4
int b_colors[b_number] = {
  ARCADA_ORANGE, 0x0F00, 0xF000, 0x00FF};
byte b_lives = 0;
byte num_lives = 5;

// WORLD
#define w_size 8
byte world[w_size][6];         // contains platforms data
int  w_color = ARCADA_YELLOW;  // Platform default color
byte w_speed = 1;
byte w_widthMin = 4;
byte w_widthMax = 8;

// ACCELEROMETER
int acc_avgX, acc_avgY, acc_avgZ;
sensors_event_t event;
```

```
#define acc_pinX 1
#define acc_pinY 2
#define acc_pinZ 3

// NeoPixels
#define RED      0xFF0000
#define YELLOW   0xFFFF00
#define GREEN    0x00FF00
#define BLACK    0x000000

void setup() {
  Serial.begin(9600);
  // initialize

  Serial.println("Hello! Arcada version of game");
  if (!arcada.arcadaBegin()) {
    Serial.print("Failed to begin");
    strip.fill(RED);
    strip.show();
    while (1);
  }

  arcada.displayBegin();         // Initialize display code
  arcada.setBacklight(0);        // Initial display off
  arcada.display->setRotation(0);         // Rotate to portrait
  arcada.display->fillScreen(background); // Clear screen if necessary
  arcada.display->setTextColor(ARCADA_GREEN, ARCADA_BLACK);
  arcada.display->setTextSize(2);
  arcada.display->println(" ");
  arcada.display->println("  Bounce!");

  if( !arcada.hasAccel() ) {
    strip.fill(YELLOW);
    strip.show();
    arcada.haltBox("An accelerometer is required for this gamne");
  }

  // Set up the logo bitmap
  int logo_origin_x = (128 - 2*LOGO_WIDTH ) / 2;
  int logo_origin_y = (160 - 2*LOGO_HEIGHT) / 2;
  for(int y=0; y<LOGO_HEIGHT; y++) {
    for(int x=0; x<LOGO_WIDTH; x++) {
      uint8_t c = logo_mask[y][x / 8];
      if(c & (0x80 >> (x & 7))) {
        int xx = logo_origin_x+2*x;
        int yy = logo_origin_y+2*y;
        arcada.display->drawPixel(xx, yy, ARCADA_WHITE);
        arcada.display->drawPixel(xx+1, yy, ARCADA_WHITE);
        arcada.display->drawPixel(xx, yy+1, ARCADA_WHITE);
        arcada.display->drawPixel(xx+1, yy+1, ARCADA_WHITE);
      }
    }
  }
  arcada.display->println(" ");
  arcada.display->println(" ");
  arcada.display->println(" ");
  arcada.display->println(" ");
  arcada.display->println(" ");
  arcada.display->setTextColor(ARCADA_ORANGE, ARCADA_BLACK);
  arcada.display->setTextSize(1);
  arcada.display->println(" ");
  arcada.display->println("  Adafruit");
  arcada.display->println("   Industries");
  for (int i=0; i<220; i++) {    // Display initial text
    arcada.setBacklight(i);
    delay(14);
  }
  arcada.setBacklight(250);
  arcada.display->setTextColor(ST7735_WHITE, ST7735_BLACK);
```

```
  arcada.display->println(" ");
  arcada.display->println(" ");
  arcada.display->println("  Press Start");
  while(!(arcada.readButtons() & ARCADA_BUTTONMASK_START)) ; // wait for start
button
  arcada.display->fillScreen(background);                              // Clear
screen

  arcada.enableSpeaker(1);   // Enable the speaker and play opening tones
  pinMode(sound_pin, OUTPUT);
  beep(sound_pin,  880, 100);
  delay(100);
  beep(sound_pin, 1060, 120);
  delay(100);
  beep(sound_pin, 1800, 100);

  // NeoPixel setup
  strip.begin();
  strip.setBrightness(40);  // not too bright
  strip.show();

  // Game setup
  randomSeed(millis());
  FillWorld();

  // Get average accelerations
  acc_avgX = acc_avg(acc_pinX);
  acc_avgY = acc_avg(acc_pinY);
  acc_avgZ = acc_avg(acc_pinZ);

  ScoreSetup();
}

void loop() {
  if(millis()-lastMillis > 20){
    lastMillis = millis();
    CheckButtons();
    MovePlayer();
    ScrollWorld();
    CollideBorders();
    CollideWorld();
    DrawWorld();
    DrawPlayer();
  }
}

int acc_avg(int pin) {
  int avg = 0;
  arcada.accel->getEvent(&event);
  switch (pin) {
    case acc_pinX:
       for(int i = 0; i < 50; i++) {
         avg += event.acceleration.x;
         // delay(10);
        }
       break;
     case acc_pinY:
       for(int i = 0; i < 50; i++) {
         avg += event.acceleration.y;
         // delay(10);
        }
       break;
     case acc_pinZ:
         for(int i = 0; i < 50; i++) {
         avg += event.acceleration.z;
         // delay(10);
        }
       break;
     }
```

```
  avg /= 50;
  return avg;
}

int acc_readX() {
  int val;
  arcada.accel->getEvent(&event);
  val = -1 * constrain((event.acceleration.y - acc_avgY)*7, -64, 64); // use Y for
orientation
  // Serial.print("Acceleration: ");                                  // joystick
is at top
  // Serial.println(val);
  return(val);
}


void CheckBonus(byte thisPlatform){
  if(b_pendingPlatform && (b_platformID == thisPlatform)){
    b_pendingPlatform = false;
  }
  if(nextBonus ){
    nextBonus--;
  }
  else {
    nextBonus = random(30, 60);
    b_pendingID = random(0, b_number);
    b_pendingPlatform = true;
    b_platformID = thisPlatform;
  }
}

void BonusReset(){
  nextBonus = random(30, 60);
  b_pendingPlatform = false;
  b_remaining = 0;
  arcada.display->fillRect(0, s_height+2, 96, 10, background);
  arcada.display->fillRect(0, s_height+1, 96, 1, w_color);
  EndUseBonus();
}

void ClearBonus(){
  EndUseBonus();
  b_remaining = 0;
  p_color = p_color_default;
  arcada.display->fillRect(0, s_height+2, 96, 10, background);
  arcada.display->fillRect(0, s_height+1, s_width-1,  1, w_color);
  b_used = false;
}

void GetBonus(){
  ClearBonus();
  b_onUseID = b_pendingID;
  b_pendingPlatform = false;
  beep(sound_pin, 3500, 50);  // SOUND
  p_color = b_colors[b_onUseID];
  arcada.display->fillRect(0, s_height+2, 96, 10, background);

  switch (b_onUseID){
  case 0:
    drawString(1, textline1, "Jet pack", b_colors[b_onUseID], 1);
    b_max = 100;
    b_remaining = b_max;
    break;
  case 1:
    drawString(1, textline1, "Frog legs", b_colors[b_onUseID], 1);
    b_max = 100;
    b_remaining = b_max;
    break;
  case 2:
```

```
    drawString(1, textline1, "Boost", b_colors[b_onUseID], 1);
    b_max = 1;
    b_remaining = b_max;
    break;
  case 3:
    drawString(1, textline1, "Low gravity", b_colors[b_onUseID], 1);
    b_max = 255;
    b_remaining = b_max;
    break;
  }
  if(b_remaining > 1) {
    arcada.display->fillRect(95, s_height+2, 1, 10, b_colors[b_onUseID]);
    arcada.display->fillRect( 0, s_height+1, 95, 1, b_colors[b_onUseID]);
  }
}

void UseBonus(){
  if(b_remaining){
    b_used = true;
    arcada.display->fillRect(b_remaining*95/b_max, s_height+2, 1, 10, background);
    arcada.display->drawPixel(b_remaining*95/b_max, s_height+1, w_color);
    b_remaining --;
    arcada.display->fillRect(b_remaining*95/b_max, s_height+2, 1, 10,
b_colors[b_onUseID]);
    arcada.display->drawPixel(b_remaining*95/b_max, s_height+1,
b_colors[b_onUseID]);

    switch (b_onUseID){
    case 0:
      if(p_vY > -1){
        p_vY -= 0.3;
      }
      beep(sound_pin, 100, 3 + abs(10 * p_vY)); // SOUND
      break;
    case 1:
      p_bounceSpeed = p_bounceSpeed_default * 1.5;
      break;
    case 2:
      p_width = min(p_width+1, p_width_default+12);
      p_height = min(p_height+1, p_height_default+12);
      b_lives = min(b_lives+1, 12);
      beep(sound_pin, 50+150*b_lives, 100);  // SOUND
      break;
    case 3:
      gravity = gravity_default * 0.3;
      break;
    }

    if(!b_remaining){
      ClearBonus();
    }
  }
}

void EndUseBonus(){
  b_used = false;

  switch (b_onUseID){
  case 0:
    break;
  case 1:
    p_bounceSpeed = p_bounceSpeed_default;
    break;
  case 2:
    break;
  case 3:
    gravity = gravity_default;
    break;
  }
```

```
}

void CheckButtons(){
  uint32_t pressed_buttons;
  pressed_buttons = arcada.readButtons();

  if(pressed_buttons & ARCADA_BUTTONMASK_START) {  // pause
    Serial.print("START");
    delay(500);
    while(!(arcada.readButtons())) {  // wait until any button is pressed again
       strip.setPixelColor(0, GREEN);
       strip.show();
       delay(100);
       strip.setPixelColor(0, BLACK);
       strip.show();
       delay(100);
    }
    pressed_buttons = 0;
    return;
  }
  if(pressed_buttons & ARCADA_BUTTONMASK_B) {  // use bonus
    Serial.print("B");
    if(b_used = true) {
      EndUseBonus();
    } else {
      UseBonus();
    }
    return;
  }
  if(pressed_buttons & ARCADA_BUTTONMASK_SELECT) {  // SELECT toggles sound on/off
    Serial.print("SELECT");
    if(sound_on) {
       sound_on = 0;
       drawChar(s_width-6, textline2, speaker_icon, ARCADA_RED);  // Put character
in lower right corner to indicate speaker off
    } else {
       sound_on = 1;
       arcada.display->fillRect( s_width-6, textline2, s_width-1, textline2+9,
background); // Blank lower right character = sound on
    }
  }
}

void DrawPlayer(){
  CheckButtons();  // check more frequently
  arcada.display->fillRect(p_lastX, p_lastY, p_width, p_height, background); //
erase previous pos
  if(p_lastX > (s_width-1 - p_width)) // if across the edge of the screen
    arcada.display->fillRect(0, p_lastY, p_lastX + p_width - s_width, p_height,
background);

  arcada.display->fillRect(p_X, p_Y, p_width, p_height, p_color); // draw new pos
  if(p_X > (s_width-1 - p_width)) // if across the edge of the screen
    arcada.display->fillRect(0, p_Y, p_X + p_width - s_width, p_height, p_color);
}

void MovePlayer(){
  p_lastX = p_X;
  p_lastY = p_Y;

  p_vX = (p_vX + acc_readX()) / 20;
  p_vY = (p_vY + gravity);

  p_X = p_X + p_vX;
  if(p_X > s_width){
    p_X = 0;
  }
  if(p_X < 0){
    p_X = s_width - p_X;
```

```
  }
  p_Y = p_Y + p_vY;

  if((p_vY < -4) && (p_color != p_color_dead)){
    beep(sound_pin, 600+p_vY*100, 25);  // SOUND
  }
  if(b_used && (b_onUseID == 3)) { // if low gravity is on use
    beep(sound_pin, 300-abs(p_vY*100), 25);  // SOUND
  }
}

void CollideBorders() {
  if(p_Y > s_height - p_height){ // touch the floor
    p_vY = -p_bounceSpeed;
    p_Y = s_height - p_height;
    if(!b_lives){
      if(p_color != p_color_dead){
        BonusReset();
        ScoreReset();
      }
    }
    else {
      arcada.display->fillRect(p_lastX, p_lastY, p_width, p_height, background);
      p_width--;
      p_height--;
      b_lives--;
      beep(sound_pin, 100+150*b_lives, 100);  // SOUND
    }
  }
}

void ScoreSetup(){
  arcada.display->fillRect(0, s_height+2, 128, 10, background);
  arcada.display->fillRect(0, s_height+1, 128, 1, w_color);
  // highscore = EEPROM.read(0);
  // highscore += (EEPROM.read(1)<<8);
  if(highscore > 64000){
    highscore = 0;
  }
  drawString(1, textline2, "High score:", w_color, 1);
  drawInt(highscore, 66, textline2, w_color, background);
}

void ScoreReset(){
  if(p_color != p_color_dead) {
    p_color = p_color_dead;
    beep(sound_pin, 50, 500); // SOUND
    if(score > highscore){
      if(score < 64000) {
        // EEPROM.write(0, score);
        // EEPROM.write(1, score>>8);
      }
      else { // bad value, zero out
        // EEPROM.write(0,0);
        // EEPROM.write(1,0);
      }
      highscore = score;
      flashMessage();
      // drawInt(score, 66, textline2, p_color_default, background);
    }
    else {
      drawString(1, textline1, "Last score:", w_color, 1);
      drawInt(score, 66, s_height+3, w_color, background);
    }
    score = 0;
    drawInt(score, 97, textline1, w_color, background);
  }
}
```

```
void flashMessage() {
   uint8_t i;
   arcada.display->fillRect( 0, textline2, s_width-10, textline2+9, background);
   for(i=0; i<5; i++) {
     drawString(1, textline2, "HIGHSCORE!", p_color_default, 1);
     delay(400);
     arcada.display->fillRect( 0, textline2, s_width-10, textline2+9, background);
     delay(400);
   }
   drawString(1, textline2, "High score:", w_color, 1);
   drawInt(highscore, 66, textline2, w_color, background);
}

void ScoreAdd(){ // add to the current score lower left
  score++;
  beep(sound_pin, 3000, 50);  // SOUND
  if(score > highscore) {
    drawInt(score, 97, textline1, p_color_default, background);
  }
  else {
    drawInt(score, 97, textline1, w_color, background);
  }
  w_widthMin = max(1, 4 - score/50);
  w_widthMax = max(2, 8 - score/50);
  if(score == 1) {
    p_color = p_color_default;
  }
}

// transform from int to string and display it
void drawInt(unsigned int num, byte nx, byte ny, unsigned int color, unsigned int
color2) {
  arcada.display->fillRect(nx, ny, 29, 7, color2);
  drawChar(nx+24, ny, 48+(num%10), color);

  if(num > 9) {
    drawChar(nx+18, ny, 48+(num%100)/10, color);

    if(num>99) {
      drawChar(nx+12, ny, 48+(num%1000)/100, color);

      if(num>999) {
        drawChar(nx+6, ny, 48+(num%10000)/1000, color);

        if(num>9999) {
          drawChar(nx, ny, 48+(num%100000)/10000, color);
        }
      }
    }
  }
}

/*
world[a][b] ->
 a ->  platform id
 b ->  0 = x
 1 = y
 2 = last x
 3 = last y
 4 = width
 4 = last width
 */

void FillWorld(){
  for(byte i=0; i<w_size; i++) {
    world[i][0] = (s_width * (w_size-i-1) / w_size)/2;
    world[i][1] = 1 + i * (s_height / w_size);
    world[i][2] = world[i][0];
    world[i][3] = world[i][1];
```

```
    world[i][4] = s_width * (i+1)/w_size;
    world[i][5] = world[i][4];
  }
}

void ScrollWorld(){
  byte fastScrolling = 0;
  if(p_Y < s_height/4) { // if the player reaches the top of the screen
    fastScrolling ++;
    if(p_Y < s_height/8) { // if the player reaches the top of the screen
      fastScrolling ++;
    }
  }
  p_Y += fastScrolling;
  for(byte i=0; i<w_size; i++) {  // for each platform
    world[i][3] = world[i][1];    // update last y
    world[i][2] = world[i][0];    // update last x
    world[i][5] = world[i][4];    // update last width
    world[i][1] += w_speed ;      // move the platform downward
    world[i][1] += fastScrolling;
    if(world[i][1] > s_height){    // if the platform reaches the bottom of the
screen
      //world[i][2] = world[i][0]; // update last x
      world[i][1] = world[i][1] - s_height + 1; // put the platform on the top of
the screen
      world[i][4] = min(random(w_widthMin, w_widthMax)*p_width_default, s_width-
world[i][0]); //pick a random width
      world[i][0] = random(0, s_width-world[i][4]); // pick a random x
      CheckBonus(i); // check if the new platform will have a bonus
    }
  }
}

void CollideWorld() {
  for(byte i=0; i<w_size; i++) {
    if(p_vY > 0){ // if player falling
      if((p_Y+p_width <= world[i][1]) && (world[i][1] < (p_Y+p_vY+p_height)) ) { //
and at the height of the platform
        if(((p_X+p_width) >= world[i][0]) && (world[i][0]+world[i][4] > (p_X)) )
{     // and does not miss it
          p_vY = -p_bounceSpeed ;
          p_Y = world[i][1] - p_height;
          ScoreAdd();
          if(b_pendingPlatform && (b_platformID == i)) {
            GetBonus();
          }
        }
      }
    }
  }
}

void DrawWorld() {
  for(byte i=0; i<w_size; i++) {
    arcada.display->fillRect(world[i][2], world[i][3], world[i][5], 1, background);
    arcada.display->fillRect(world[i][0], world[i][1], world[i][4], 1, w_color);
  }
  if(b_pendingPlatform){
    arcada.display->fillRect(world[b_platformID][0], world[b_platformID][1],
world[b_platformID][4], 1, b_colors[b_pendingID]);
  }
}


void drawString(byte x, byte y, char *text, uint16_t color, bool wrap) { //
replicate tft.drawString
  arcada.display->setCursor(x,y);
  arcada.display->setTextColor(color);
  arcada.display->setTextWrap(wrap);
```

```
  arcada.display->print(text);
}

void drawChar(byte x, byte y, char text, uint16_t color) { // replicate tft.drawChar
  arcada.display->setCursor(x,y);
  arcada.display->setTextColor(color);
  arcada.display->print(text);
  Serial.println(text);
}

// A sound-producing function
void beep(unsigned char speakerPin, int frequencyInHertz, long timeInMilliseconds) {
  // http://web.media.mit.edu/~leah/LilyPad/07_sound_code.html
  int  x;
  if( sound_on ) {     // Only play if the sound_on flag is 1
    long delayAmount = (long)(1000000 / frequencyInHertz);
    long loopTime = (long)((timeInMilliseconds * 1000) / (delayAmount * 2));
    for (x = 0; x < loopTime; x++) {
      digitalWrite(speakerPin, HIGH);
      delayMicroseconds(delayAmount);
      digitalWrite(speakerPin, LOW);
      delayMicroseconds(delayAmount);
    }
  }
}
```

```
// SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*!
 * @file logo.h
 *
 * Header file to accompany the demo-3-logo example.
 * A 40x40 pixel Adafruit logo in grayscale (for rendering on LED matrix)
 * and corresponding bitmap (for PixelDust collision detection).
 * Latter has the 'seeds' filled so randomly-placed grains don't
 * end up trapped in there.
 *
 */

#define LOGO_WIDTH  40
#define LOGO_HEIGHT 40

const uint8_t logo_gray[LOGO_HEIGHT][LOGO_WIDTH] = {
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X16, 0XD2, 0XEB, 0X39,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X01,
  0XB7, 0XFF, 0XFF, 0XB1, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X71, 0XFF, 0XFF, 0XFF, 0XF7, 0X0D, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X30, 0XF8, 0XFF, 0XFF, 0XFF, 0XFF,
  0X57, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X0A, 0XD6, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XA9, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X98, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF2, 0X08, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X51, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
```

```
0XFF, 0X4D, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X0D, 0XE8, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X9F, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X72,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XEC, 0X02, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0XCB, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0X25, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X22, 0X70, 0X78, 0X77, 0X76, 0X75, 0X74, 0X73,
0X72, 0X6C, 0X4D, 0X15, 0X00, 0X00, 0X04, 0XFB, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X3F, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0XDA, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFC, 0XB3, 0X2D, 0X0C, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X3C, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0XDD, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XF8, 0X52, 0XEB, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFE, 0X17, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X46, 0XFD, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF6, 0XC9, 0XFF, 0XFF, 0XFF, 0XC8,
0X9F, 0XFF, 0XFF, 0XFF, 0XFF, 0XC4, 0X00, 0X17, 0X3D, 0X41, 0X27, 0X03,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X89, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFA,
0XFF, 0XFF, 0XFD, 0X22, 0X0F, 0XFF, 0XFF, 0XFF, 0XFC, 0X76, 0XC4, 0XFE,
0XFF, 0XFF, 0XFF, 0XEE, 0XA2, 0X50, 0X0A, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X05, 0XC7, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XC9, 0X00, 0X00, 0XFA, 0XFF, 0XFF,
0XE6, 0XFC, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF4, 0XAD,
0X5C, 0X10, 0X00, 0X00, 0X00, 0X00, 0X21, 0XEF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XA0, 0X00,
0X16, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF9, 0XB7, 0X3D, 0X00, 0X00, 0X00, 0X55,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X68, 0X2B, 0X6A, 0XD9,
0XFF, 0XFF, 0X9F, 0X00, 0X58, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XEC,
0X00, 0X00, 0X00, 0X00, 0X98, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0X94, 0X00, 0X00, 0X06, 0X91, 0XFF, 0XDC, 0X19, 0XD2, 0XFF, 0XFF, 0XFF,
0XFC, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XCF, 0X00, 0X00, 0X00, 0X00, 0X09, 0XC6, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XA9, 0X1D, 0X00, 0X00, 0XC6, 0XFF, 0XFF,
0XFF, 0XD5, 0X5C, 0X18, 0X00, 0X10, 0X9D, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XB4, 0X14, 0X00, 0X00, 0X00, 0X00,
0X00, 0X0F, 0XB9, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFB, 0XBC,
0X9E, 0XF0, 0XFF, 0XFF, 0XFF, 0X1D, 0X00, 0X00, 0X00, 0X21, 0XC7, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF7, 0X6D, 0X01, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X02, 0X65, 0XD7, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XDC, 0X9E, 0X9E,
0XC7, 0XFC, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XD4,
0X2E, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X39, 0XAF, 0XEF, 0XFF, 0XFF, 0XFF, 0XE5, 0X64, 0X6C, 0XFF, 0XF2,
0XC7, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0X97, 0X09, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X69, 0XF8, 0XFF, 0XFF, 0XFF, 0XFF, 0XD3, 0X18,
0X00, 0X5E, 0XFF, 0XA2, 0X00, 0X8E, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XFF, 0XFF, 0XFF, 0XE7, 0X50, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X64, 0XFF, 0XFF, 0XFF, 0XFF,
0XFF, 0XF3, 0X20, 0X00, 0X13, 0XE1, 0XFF, 0XC0, 0X00, 0X05, 0XD5, 0XFF,
0XFF, 0XFB, 0XC9, 0XEA, 0XFF, 0XFB, 0XCB, 0X72, 0X0D, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X21, 0XF5,
0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X97, 0X00, 0X1A, 0XCC, 0XFF, 0XFF, 0XFC,
0X20, 0X00, 0X65, 0XFF, 0XFF, 0XFF, 0XF7, 0X55, 0X0C, 0X04, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X9C, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XCE, 0X8F, 0XEF,
0XFF, 0XFF, 0XFF, 0XFF, 0XAD, 0X00, 0X25, 0XFF, 0XFF, 0XFF, 0XFF, 0XF9,
0X31, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X0A, 0XF4, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
```

```
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X96, 0X60, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XB9, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X52, 0XFF, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF0, 0XFF, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFE, 0X1A, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0XA6, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0XB1, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0X54, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X08, 0XF2, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFB, 0X3A, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X74, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X4F, 0XFF, 0XFF, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X76, 0X00, 0XD1, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X7A, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0XA3, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XF9, 0X71,
  0X00, 0X00, 0X5F, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0X7B, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X07, 0XF0, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XF9, 0XA8, 0X2A, 0X00, 0X00, 0X00, 0X01, 0XB5, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X7D, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X42, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0XF8, 0XB3, 0X5F, 0X10, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X0D,
  0XC3, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X7E, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X31,
  0XFC, 0XFF, 0XF8, 0XB1, 0X5D, 0X0F, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X08, 0X94, 0XFE, 0XFF, 0XFF, 0XFF, 0XFF, 0XFF,
  0XFF, 0X80, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X37, 0X4F, 0X0E, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X50, 0XED,
  0XFF, 0XFF, 0XFF, 0XFF, 0XFF, 0X81, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X1E, 0XC3, 0XFF, 0XFF, 0XFF, 0XFF, 0X82, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X04, 0X83, 0XFC, 0XFF,
  0XFF, 0X7C, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X40, 0XD9, 0XDF, 0X29, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
  0X00, 0X00, 0X00, 0X00 };
const uint8_t logo_mask[LOGO_HEIGHT][(LOGO_WIDTH + 7) / 8] = {
  0X00, 0X00, 0X06, 0X00, 0X00, 0X00, 0X00, 0X0F, 0X00, 0X00, 0X00, 0X00,
  0X0F, 0X00, 0X00, 0X00, 0X00, 0X1F, 0X00, 0X00, 0X00, 0X00, 0X3F, 0X80,
  0X00, 0X00, 0X00, 0X7F, 0X80, 0X00, 0X00, 0X00, 0X7F, 0X80, 0X00, 0X00,
  0X00, 0XFF, 0XC0, 0X00, 0X00, 0X00, 0XFF, 0XC0, 0X00, 0X00, 0X01, 0XFF,
  0XC0, 0X00, 0X00, 0X01, 0XFF, 0XC0, 0X00, 0XFF, 0XF9, 0XFF, 0XC0, 0X00,
  0XFF, 0XFD, 0XFF, 0XC0, 0X00, 0X7F, 0XFF, 0XFF, 0XC0, 0X00, 0X7F, 0XFF,
  0XFF, 0XFF, 0X80, 0X3F, 0XFF, 0XFF, 0XFF, 0XF0, 0X1F, 0XFF, 0XFF, 0XFF,
  0XFE, 0X0F, 0XFF, 0XFF, 0XFF, 0XFF, 0X0F, 0XFF, 0XFF, 0XFF, 0XFF, 0X07,
  0XFF, 0XFF, 0XFF, 0XFE, 0X03, 0XFF, 0XFF, 0XFF, 0XF8, 0X00, 0XFF, 0XFF,
  0XFF, 0XF0, 0X00, 0X3F, 0XFF, 0XFF, 0XE0, 0X00, 0X7F, 0XFF, 0XFF, 0X80,
  0X00, 0XFF, 0XFF, 0XFE, 0X00, 0X01, 0XFF, 0XFF, 0XE0, 0X00, 0X03, 0XFF,
  0XFF, 0XF0, 0X00, 0X03, 0XFF, 0XFF, 0XF8, 0X00, 0X03, 0XFF, 0XFF, 0XF8,
  0X00, 0X07, 0XFF, 0XFF, 0XF8, 0X00, 0X07, 0XFF, 0XBF, 0XF8, 0X00, 0X07,
  0XFF, 0X3F, 0XF8, 0X00, 0X0F, 0XFE, 0X1F, 0XF8, 0X00, 0X0F, 0XFC, 0X1F,
  0XF8, 0X00, 0X0F, 0XE0, 0X0F, 0XF8, 0X00, 0X0F, 0X00, 0X07, 0XFC, 0X00,
  0X00, 0X00, 0X01, 0XFC, 0X00, 0X00, 0X00, 0X00, 0XFC, 0X00, 0X00, 0X00,
  0X00, 0X78, 0X00, 0X00, 0X00, 0X00, 0X18, 0X00 };
```

Gameplay

The game is played in portrait mode so you'll turn your PyGamer or PyBadge 90 degrees.

There is an introductory screen - press **START** to continue.

Basics

The display consists of a bouncing ball which bounces when it hits a flat surface (it can pass through the bottom of platforms but bounces on the tops of the platforms).

The goal is to have as many platform bounces as you can without the ball hitting the floor.

You tilt the board left and right so move the ball side to side to get on the platform you want to bounce off of. Currently tilting up and down does not activate anything.

The ball will move off the screen to the right or left to the other side - handy if the platform you want to bounce off of is too far away for a cross-screen bounce.

When the ball is yellow, it is in play and the count of bounces is on the lower right on the screen. After one play, there is a last game score below the ground level and a high score count at the bottom of the display.

You can pause the game by pressing the **START** button. The screen will freeze. Pressing any other button will start the game back up.

Sound

The default is sound on. You can use the **SELECT** button to toggle sound off and on. A red ø symbol is displayed in the lower right of the screen if the sound is muted. The buttons are not debounced - if you press **SELECT** and it still is muted, keep pressing until it is unmuted.

Bonuses

There are 4 types of bonuses that are triggered when hitting a colored platform. Use the A button to activate the bonus. The type of bonus is listed at the bottom of the display and only lasts a short time.

Going Further

There is a number of ways to extend the game in code. If top to bottom movements are detected, it could increase or decrease gravity some. If moving offscreen to the other side isn't your cup of tea, you can detect when the ball hits the bounds and bounce back at the same speed. There are still lots of buttons and NeoPixels not in use for input and output.