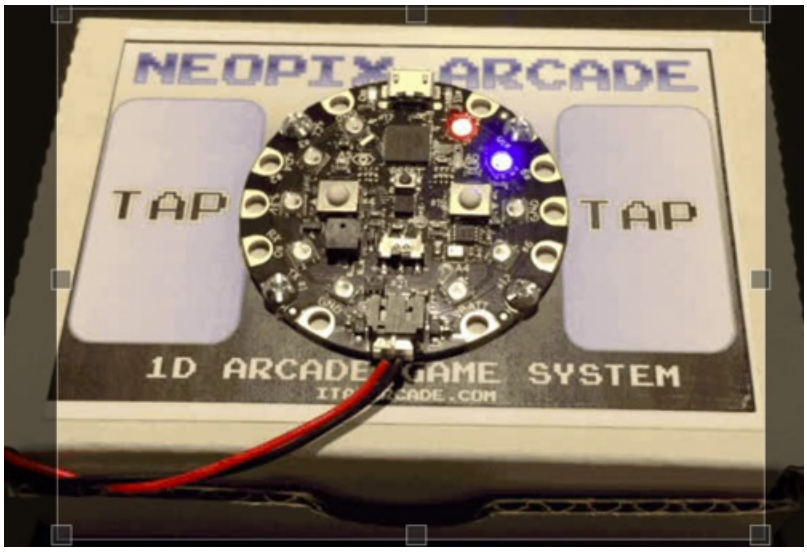
adafruit learning system

NeoPix Arcade Kit - 1D Arcade Game System - Circuit Playground

Created by itaparcade



Last updated on 2018-08-22 03:58:35 PM UTC

Guide Contents

Why create a 1D Arcade Game System?

Imagine having the ability to create retro type LED games similar to the classic handheld LED games of the late 70s and early 80s but with only 10 NeoPixels oriented in an arc combined with lots of sensors! We are challenging makers to create games that transform the Circuit Playground board into a retro handheld 1D arcade game system. The Circuit Playground board is a perfect board to explore coding based on creating retro LED type games with young programmers.

We seek to create 1D arcade games centered around retro LED gaming with just 10 NeoPixels. We focus on LED gaming because it provides a great way to teach basic coding and gaming concepts while keeping costs low. LED based games were fun as portable handheld gaming systems for us who grew up in the 80s, so I am seeking to capture that same kind of fun with Circuit Playground while teaching young programmers how to code. With all of the sensors on the Circuit Playground, it makes the perfect platform for developing LED based games. Instead of 80s LEDs with no extra sensors, Circuit Playground with NeoPixels and a variety of sensors are available to create this 1D arcade game system.

Introducing the NeoPix Arcade Kit

The NeoPix Arcade Kit is a 1D arcade game system to encourage young programmers to code. The NeoPix Arcade Kit comes with or without a preprogrammed Circuit Playground (includes the 1D Pong Game) to immediately start exploring basic game programming concepts on the Circuit Playground. The Circuit Playground Board has a host of sensors to create a variety of handheld electronic gaming projects.

The NeoPix Arcade Kit (https://adafru.it/tGB) requires no soldering and includes:

- 1 x Circuit Playground Board (preprogrammed with 1D Pong Game) (optional purchase)
- 1 x USB cable for programming
- 1 x NeoPix Arcade Box (just about any cardboard box)
- 1 x Battery Holder (3 AAA batteries not included)
- 4 x Screws for mounting Circuit Playground to Box

Even if you don't purchase our kit you can still explore 1D arcade game coding with your own Circuit Playground because the Circuit Playground 1D Pong Code is included in this article. Our code leverages Arduino multi-tasking principles to create a handheld electronic game like experience on the Circuit Playground with sound, lights, and sensor input.

Learn

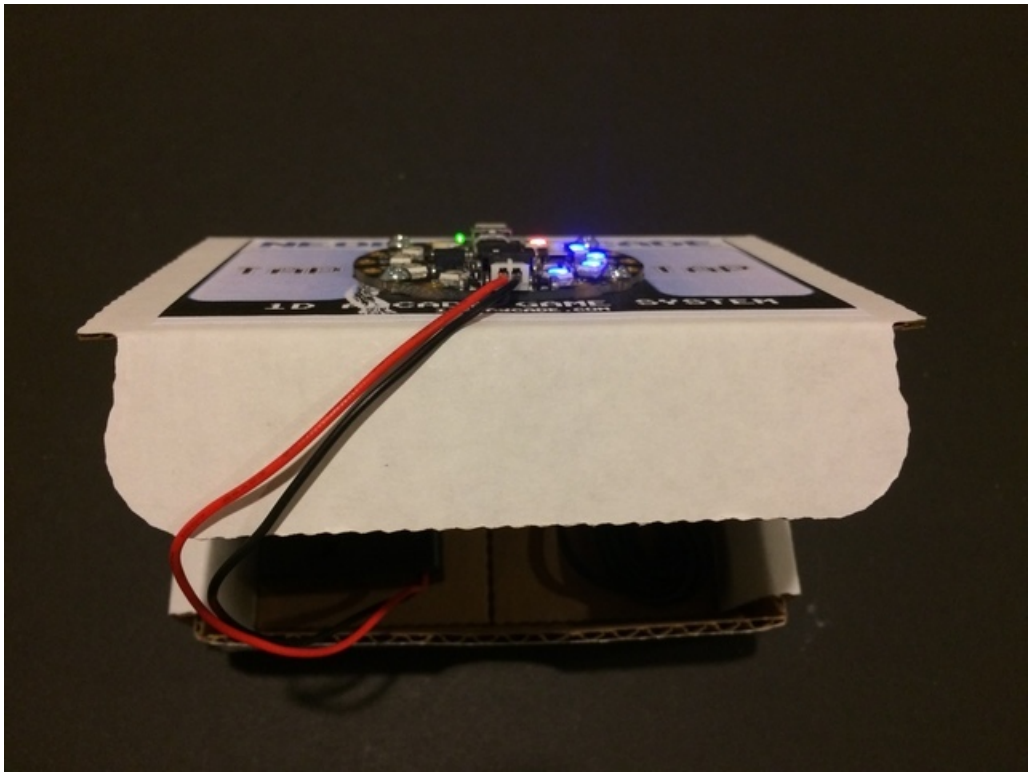
I created the NeoPix Arcade Kit to challenge makers to design 1D arcade games and explore basic game programming concepts on the Circuit Playground with young programmers. We are currently creating games to explore a variety of coding concepts with respect to 1D arcade game programming leveraging sensors on the Circuit Playground. The first game we developed for this 1D game system is a NeoPix 1D pong game to illustrate what type of games can be created on the Circuit Playground when multi-tasking different elements of an arcade game (sound, display, controller input). A great tutorial series already exists about multi-tasking on the Arduino. We leveraged the same multi-tasking coding principles to create the 1D pong game. Check out this multi-tasking tutorial (https://adafru.it/mEf) before looking at our 1D pong game code.

Build

Since we wanted to keep the costs at a minimum we decided to build the NeoPix Arcade Kit around a cardboard box. The cardboard box has dual use. First, it serves to hold the Circuit Playground, USB cable, batteries, and the screws during transport (if desired). Secondly, it allows the mounting of the Circuit Playground on the box so tap detection based games on the NeoPix arcade work consistently. In fact, the 1D Pong Game works both with tap detection and with the left and right buttons on the Circuit Playground.

The NeoPix Arcade Kit comes with a AAA battery pack, a micro USB cable, screws, a NeoPix arcade box, and a preprogrammed Circuit Playground. This was designed to make assembly as easy as possible. The following pictures show mounting process of the Circuit Playground.

Note that the bolts can be mounted in a reverse orientation to easily allow external switches to be connected to the Circuit Playground via alligator clips.




NEOPIX ARCADE
TAP
TAP
1D ARCADE GAME SYSTEM

Program

We have included the 1D Pong code so you can experience the 1D arcade gaming experience even if you don't have the NeoPix Arcade kit. We have also included the pitches.h file as well below the NeoPix Arcade code. Make sure to include this in your Arduino project. Check out this tutorial (https://adafru.it/tGC) to learn about the Circuit Playground and programming it.

```
/*************************************************************************
    NeoPix Arcade - 1D Pong Game:
    Copyright (c) 2017 iTapArcade, LLC

    Date Created: 3 December 2016
    Date Modified: 15 January 2017
    Version: 1.0
    Visit us at iTapArcade.com
    Follow us on twitter @iTapArcade

    Support this project by buying NeoPix Arcade Kits from iTapArcade.com at
    http://itaparcade.com/collections/arcade-interface-modules/products/neopix-arcade-kit-1d-arcade-game-

    Playing the Game:
    - 2 button 1D Pong Game for the Circuit Playground Developer Edition Board
    - 1 or 2 Players
    Controls:
    - Press corresponding button (left or right) at the correct time when ball is on your side to hit bal
    - When using tap capability to play, tap on the NeoPix Arcade Box (or on a table) when the ball is on
    Gameplay:
    - 1 Player: See how long you can volley back and forth
    - 2 Players: Each person gets a button.  After 5 misses the other player wins. The misses from each p

    Purpose:
    - This code was created to inspire makers from 5 to 99 to be creative at making 1D arcade games on th
    - This NeoPix Arcade 1D Pong Game should be thought of as the "hello world" code to help inspire teac
    - The background sound and game controls (buttons and tap detection) were coded in a way to suport mu
    - Think of the NeoPixels as the display ("like TV video game display") that can be programmed to show
    - Leverage the sensors as "controller" inputs to support gameplay
    - Leverage the buzzer to be creative with your game sounds like the early 80s portable electronic gam
    - Making 1D arcade games on the Circuit Playground should challenge the way you think about game prog
    - Have fun creating and making your own 1D arcade games and Tweet us @iTapArcade what you create so w

    Support this project by buying NeoPix Arcade Kits from iTapArcade.com at
    http://itaparcade.com/collections/arcade-interface-modules/products/neopix-arcade-kit-1d-arcade-game-

 ************************/

#include <Adafruit_CircuitPlayground.h>
#include "pitches.h"

#define CLICKTHRESHHOLD 8

unsigned long previousMillis = 0;
const long interval[] = {25, 15, 10, 6};        // interval at which to scan neopixels per game_level (mi
int neo_pixel_number = 0;
int game_level = 0;

boolean ledState = LOW;
boolean tap_detect_reset_timer = LOW;
boolean directionState = LOW;
```

```
boolean just_started = HIGH;
boolean game_over = HIGH;
int misses_player_right = 0;
int misses_player_left = 0;
int hits_in_a_row = 0;
int max_hits_in_a_row_per_game_level = 8;
bool leftButtonPressed;
bool rightButtonPressed;
bool lockout_rightButton = LOW;
bool lockout_leftButton = LOW;
bool mute = LOW;
bool sound_enable = HIGH;
const int brightness = 5;              // Change this value to change the NeoPixel brightness

/*----Background Music Melody for 1D Pong Game----*/
const int numNotes = 3;                     // number of notes we are playing
int melody[] = {                            // specific notes in the melody
  NOTE_C2, NOTE_C3, NOTE_C2
};

int noteDurations[] = {     // note durations: 4 = quarter note, 8 = eighth note, etc.:
  8, 8, 8
};
/*----------------------------------------*/

int ledPin;
long background_sound_on_time;     // milliseconds of on-time
long background_sound_offtime;    // milliseconds of off-time


int background_sound_select;
unsigned long previousMillis_background_sound;

int thisNote;
int noteDuration;
int pauseBetweenNotes;
boolean enable_background_sound = LOW;

int Detected_Tap = LOW;
unsigned long previousMillis_tap_detector = 0;

// constants won't change :
const long tap_detect_reset_duration = 100;      // interval at which to reset tap detection (millisecond

void Update_Tap_Detector()
{
  if (Detected_Tap)
  {
    unsigned long currentMillis_tap_detector = millis();

    if (currentMillis_tap_detector - previousMillis_tap_detector >= tap_detect_reset_duration) {

      previousMillis_tap_detector = currentMillis_tap_detector;

      // if the timer has not started turn it on to reset tap detection
      if (tap_detect_reset_timer == LOW) {
        tap_detect_reset_timer = HIGH;
      } else {
        tap_detect_reset_timer = LOW;
        Detected_Tap = LOW;
```

```
      }
    }

    digitalWrite(13, Detected_Tap);
  }
}

void Update_Background_Sound()
{
  if (enable_background_sound)
  {
    unsigned long currentMillis_background_sound = millis();

    if ((background_sound_select == HIGH) && (currentMillis_background_sound - previousMillis_background_
    {
      // update counter to select next background_sound_on_time
      background_sound_select = LOW;  // Turn it off
      previousMillis_background_sound = currentMillis_background_sound;  // Remember the time

      if (thisNote < numNotes - 1)
      {
        thisNote = thisNote + 1;
        background_sound_offtime = 0;
      }
      else {
        thisNote = 0;
        background_sound_offtime = 0;
      }

    }
    else if ((background_sound_select == LOW) && (currentMillis_background_sound - previousMillis_backgro
    {
      noteDuration = 1000 / noteDurations[thisNote];
      CircuitPlayground.playTone(melody[thisNote] * (game_level + 1), noteDuration);
      pauseBetweenNotes = noteDuration * 1.30;//1.30;
      background_sound_on_time = pauseBetweenNotes;
      background_sound_select = HIGH;  // turn it on
      previousMillis_background_sound = currentMillis_background_sound;
    }
  }
}

void TapDetector(void) {
  Detected_Tap = HIGH;
}

void setup() {

  /*-----Initialize Variables for Game ------*/
  CircuitPlayground.begin();
  CircuitPlayground.setBrightness(brightness);
  CircuitPlayground.clearPixels();
  CircuitPlayground.setAccelRange(LIS3DH_RANGE_2_G);   // 2, 4, 8 or 16 G!
  CircuitPlayground.setAccelTap(1, CLICKTHRESHHOLD);

  attachInterrupt(digitalPinToInterrupt(7), TapDetector, FALLING);

  thisNote = 0;
  noteDuration = 0;
```

```
  pauseBetweenNotes = 0;

  ledPin = 13;
  pinMode(ledPin, OUTPUT);

  background_sound_on_time = 0;//was 350
  background_sound_offtime = 0;

  background_sound_select = LOW;
  previousMillis_background_sound = 0;

  digitalWrite(13, Detected_Tap);
}

void Show_Intro(void) {

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval[0]) {
    // save the last time you turned on NeoPixel
    previousMillis = currentMillis;

    if (ledState == LOW) {
      ledState = HIGH;
      if (neo_pixel_number == 9 || neo_pixel_number == 0)
      {
        CircuitPlayground.clearPixels();
        CircuitPlayground.setPixelColor(neo_pixel_number,   255,  0, 0);
      } else {
        CircuitPlayground.setPixelColor(neo_pixel_number,   0,   0, 255);
      }
      if (directionState == LOW) {
        if (neo_pixel_number < 9) {
          neo_pixel_number = neo_pixel_number + 1;
        } else {
          //  neo_pixel_number = 0;
          neo_pixel_number = neo_pixel_number - 1;
          directionState = HIGH;
        }
      } else {
        if (neo_pixel_number > 0) {
          neo_pixel_number = neo_pixel_number - 1;
        } else {
          neo_pixel_number = neo_pixel_number + 1;
          directionState = LOW;
        }
      }
    } else {
      ledState = LOW;
    }
  }
}

void Update_Game_State(void) {

  // Show Intro Display if Game Over

  if (game_over) Show_Intro();

  // Check if either player is ready to start game:
```

```
  if ((CircuitPlayground.leftButton() && game_over) || (CircuitPlayground.rightButton() && game_over))
  {
    CircuitPlayground.clearPixels();
    game_over = LOW;
    enable_background_sound = HIGH;
    misses_player_left = 0;
    misses_player_right = 0;
    game_level = 0;
    neo_pixel_number = 1;
    directionState = LOW;

    // Starting Melody for Game
    delay(50);
    CircuitPlayground.playTone(900, 100);
    delay(50);
    CircuitPlayground.playTone(600, 100);
    delay(50);
    CircuitPlayground.playTone(900, 100);
    delay(50);
    CircuitPlayground.playTone(600, 100);
    delay(500);
  }

  if (!game_over) play_game();

}

void play_game(void) {

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval[game_level]) {
    // save the last time you turned on NeoPixel
    previousMillis = currentMillis;
    // check if any buttons have been released
    if (!CircuitPlayground.rightButton()) lockout_rightButton = LOW;
    if (!CircuitPlayground.leftButton()) lockout_leftButton = LOW;

    if (CircuitPlayground.rightButton() && neo_pixel_number < 9) lockout_rightButton = HIGH;
    if (CircuitPlayground.leftButton() && neo_pixel_number > 0) lockout_leftButton = HIGH;

    if (ledState == LOW) {
      ledState = HIGH;
      if (neo_pixel_number == 9)
      {
        CircuitPlayground.clearPixels();

        if ((CircuitPlayground.rightButton() && !lockout_rightButton) || Detected_Tap) {
          lockout_rightButton = HIGH;
          CircuitPlayground.setPixelColor(neo_pixel_number,   0,   255, 0);
          CircuitPlayground.playTone(600, 100);
          hits_in_a_row = hits_in_a_row + 1;
          if (hits_in_a_row == max_hits_in_a_row_per_game_level) {
            hits_in_a_row = 0;
            if (game_level < 3) {
              game_level = game_level + 1;
              delay(50);
              CircuitPlayground.playTone(900, 100);
```

```
          delay(50);
          CircuitPlayground.playTone(900, 100);
          delay(50);
          CircuitPlayground.playTone(900, 100);
        } else {
          game_level = 0;
        }
      }

    } else
    {
      CircuitPlayground.setPixelColor(neo_pixel_number,   255,  0, 0);
      CircuitPlayground.playTone(300, 100);
      misses_player_right = misses_player_right + 1;
      hits_in_a_row = 0;
      update_score();
    }
  } else if (neo_pixel_number == 0)
  {
    CircuitPlayground.clearPixels();
    if ((CircuitPlayground.leftButton() && !lockout_leftButton) || Detected_Tap) {
      lockout_leftButton = HIGH;
      CircuitPlayground.setPixelColor(neo_pixel_number,   0,   255, 0);
      CircuitPlayground.playTone(600, 100);
      hits_in_a_row = hits_in_a_row + 1;
      if (hits_in_a_row == max_hits_in_a_row_per_game_level) {
        hits_in_a_row = 0;

        if (game_level < 3) {
          game_level = game_level + 1;
          delay(50);
          CircuitPlayground.playTone(900, 100);
          delay(50);
          CircuitPlayground.playTone(900, 100);
          delay(50);
          CircuitPlayground.playTone(900, 100);
        } else {
          game_level = 0;
        }

      }
    } else
    {
      CircuitPlayground.setPixelColor(neo_pixel_number,   255,  0, 0);
      CircuitPlayground.playTone(300, 100);
      misses_player_left = misses_player_left + 1;
      hits_in_a_row = 0;
      update_score();
    }
  } else {
    if (game_level == 0) CircuitPlayground.setPixelColor(neo_pixel_number,   0,   0, 255);
    if (game_level == 1) CircuitPlayground.setPixelColor(neo_pixel_number,   255,   255, 0);
    if (game_level == 2) CircuitPlayground.setPixelColor(neo_pixel_number,   0,   255, 255);
    if (game_level == 3) CircuitPlayground.setPixelColor(neo_pixel_number,   255,   255, 255);
  }
  if (directionState == LOW) {
    if (neo_pixel_number < 9) {
      neo_pixel_number = neo_pixel_number + 1;
    } else {
      //  neo_pixel_number = 0;
```

```
          neo_pixel_number = neo_pixel_number - 1;
          directionState = HIGH;
        }
      } else {
        if (neo_pixel_number > 0) {
          neo_pixel_number = neo_pixel_number - 1;
        } else {
          neo_pixel_number = neo_pixel_number + 1;
          directionState = LOW;
        }
      }
    } else {
      ledState = LOW;
    }
  }
}


void update_score(void)
{
  // Show misses from player right
  for (int i = 0; i < misses_player_right; ++i) CircuitPlayground.setPixelColor(9 - i,   255,   0, 0);
  if (misses_player_right == 5) {
    // game over for right player
    // add some sound and visuals for winner
    delay(50);
    CircuitPlayground.playTone(300, 100);
    delay(50);
    CircuitPlayground.playTone(300, 100);
    delay(50);
    CircuitPlayground.playTone(300, 100);
  }
  // Show misses from player left
  for (int i = 0; i < misses_player_left; ++i) CircuitPlayground.setPixelColor(i,   255,   0, 0);
  if (misses_player_left == 5) {
    // game over for left player
    // add some sound and visuals for winner
    delay(50);
    CircuitPlayground.playTone(300, 100);
    delay(50);
    CircuitPlayground.playTone(300, 100);
    delay(50);
    CircuitPlayground.playTone(300, 100);
  }
  delay(3000);
  if (misses_player_right == 5) {
    game_over_player_right();
  }
  else if (misses_player_left == 5) {
    game_over_player_left();
  } else
  {
    CircuitPlayground.clearPixels();
  }
}


void game_over_player_right(void) {
```

```
  // game over for right player
  for (int i = 0; i < misses_player_left; ++i) CircuitPlayground.setPixelColor(i,   0,   255, 0);
  delay(50);
  if (CircuitPlayground.slideSwitch()) CircuitPlayground.playTone(300, 100);
  delay(50);
  if (CircuitPlayground.slideSwitch()) CircuitPlayground.playTone(600, 100);
  delay(50);
  if (CircuitPlayground.slideSwitch()) CircuitPlayground.playTone(900, 100);
  delay(5000);
  game_over = HIGH;
  enable_background_sound = LOW;
  CircuitPlayground.clearPixels();

}

void game_over_player_left(void) {
  // game over for left player
  for (int i = 0; i < misses_player_right; ++i) CircuitPlayground.setPixelColor(9 - i,   0,   255, 0);

  delay(50);
  CircuitPlayground.playTone(300, 100);
  delay(50);
  CircuitPlayground.playTone(600, 100);
  delay(50);
  CircuitPlayground.playTone(900, 100);
  delay(5000);
  game_over = HIGH;
  enable_background_sound = LOW;
  CircuitPlayground.clearPixels();
}

void loop() {

  Update_Background_Sound();
  Update_Tap_Detector();
  Update_Game_State();

}
```

```
/*************************************************************
 * Musical Notes via https://www.arduino.cc/en/Tutorial/ToneMelody  *
 *************************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
```

```
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
```

```
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Play

The NeoPix Arcade Kit is a 1D arcade game system that comes with a preprogrammed Circuit Playground that includes our 1D Pong Game to immediately help young programmers start exploring basic game programming concepts on the Circuit Playground. Where else can you find a 1D arcade game system to motivate young coders while having fun? This is the first of its kind! Follow us on Twitter @iTapArcade and tell others about this project. Check out the NeoPix Arcade Kit (https://adafru.it/tGB) assembled and in action below.