

# マイクロビットの プログラム例

塾のフリータイムや自宅であそべるよう、マイクロビットのプログラム例をいくつか用意しました。

いずれもマイクロビットだけであそべるものです。

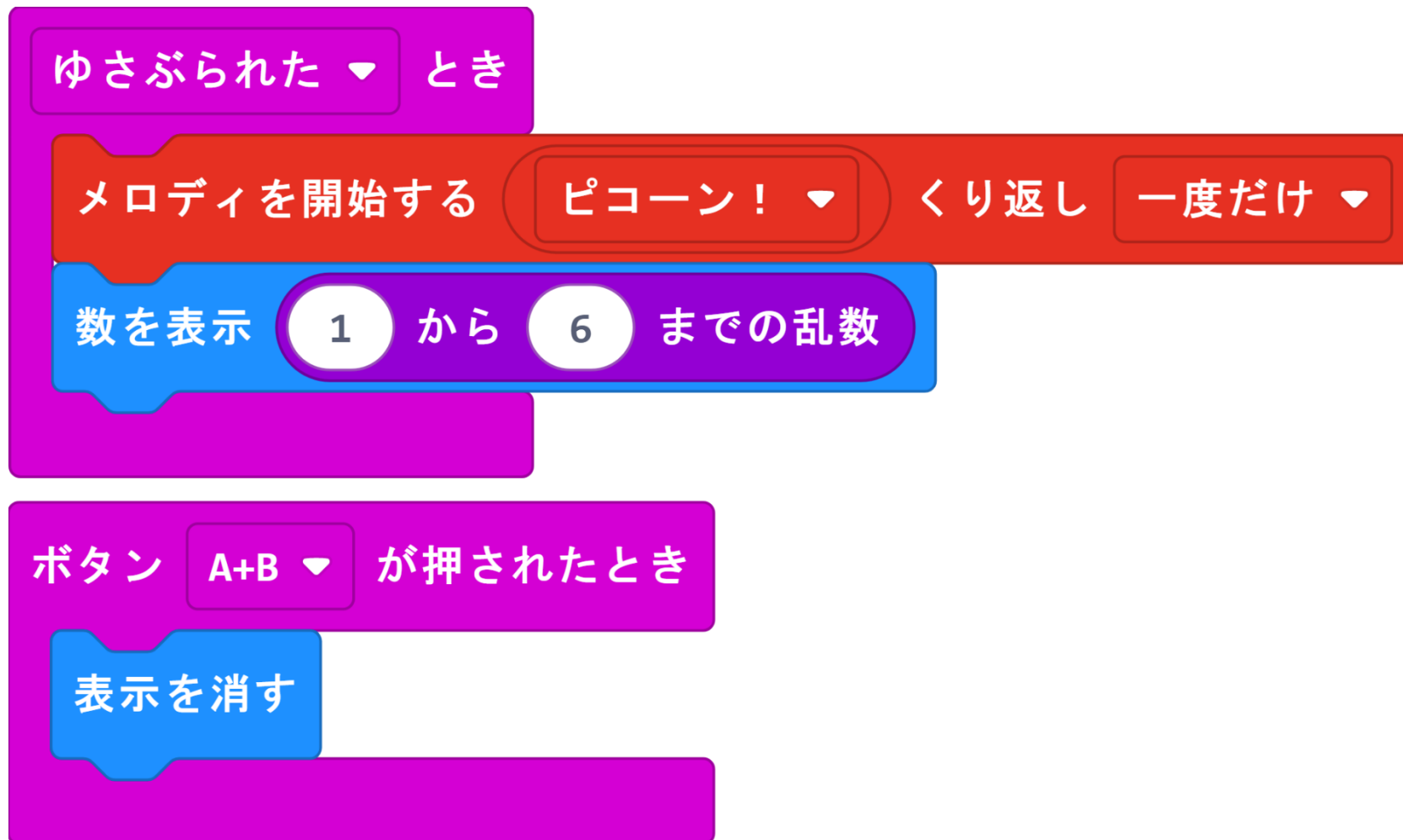
好きなものを選んでプログラムしてみてください。

ゆめほたる環境科学技術クラブ

# サイコロ

- マイクロビットをふるると「ピコーン！」という音がなり、LEDディスプレイに、1～6のうち、ランダムな数字が表示されます
- [A]ボタンと[B]ボタンを同時におすと、表示が消えます

# サイコロ (プログラム)



# 楽器

- [A]ボタンをおしている間だけ、音が鳴ります
- マイクロビットのかたむき（横方向）により、音の高さが変わります

# 楽器（説明）

- 音を鳴らすために「音を鳴らす 高さ (Hz)」ブロックを使います
- マイクロビットが水平に置かれているとき、「傾斜 (°) ロール」は「0」、左向きに垂直に立てたとき「-90」、右向きに垂直に立てたとき「90」になります
- 「 $(\text{傾斜} + 90) \times 5$ 」という計算の結果で音を鳴らすようにすれば、左向きに立てたとき「0」、水平のとき「450」、右向きに立てたとき「900」になります
- 真ん中のラが440Hzなので、マイクロビットを水平にしたときに真ん中のラに近い音が鳴ります

# 楽器 (プログラム)

ずっと

もし ボタン A ▼ が押されている なら

音を鳴らす 高さ (Hz) 傾斜 (°) ローラ ▼ + ▼ 90 × ▼ 5

でなければ -

すべての音を停止

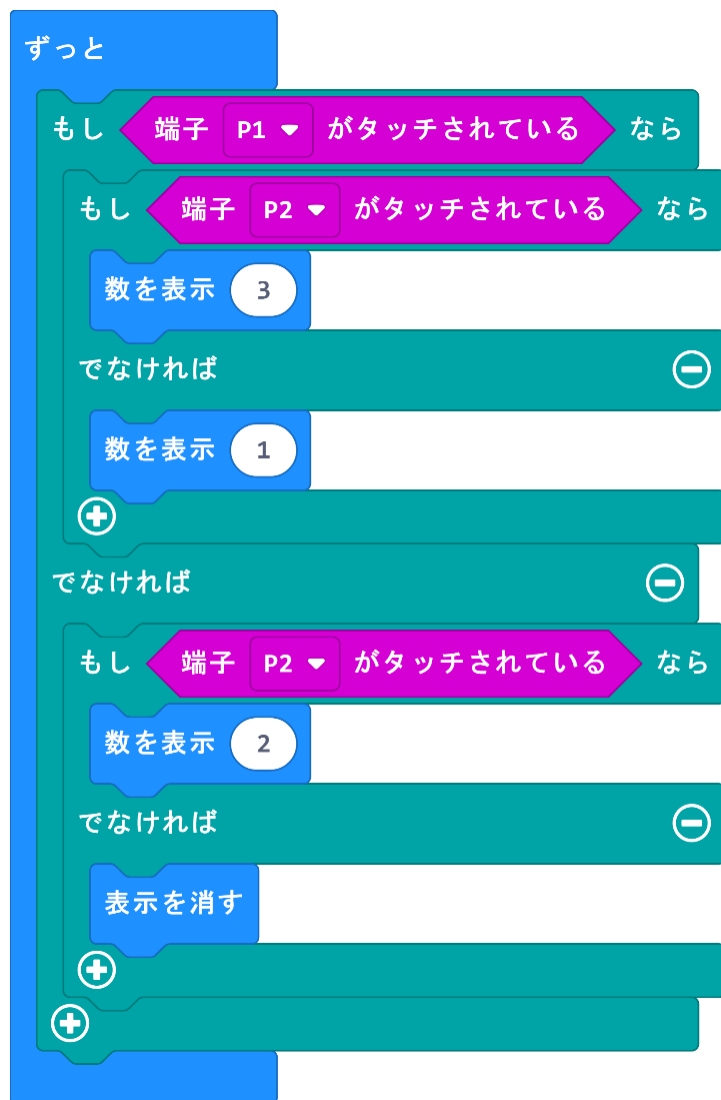
+

Detailed description: This is a Scratch-style programming block for a musical instrument. It starts with a blue 'ずっと' (Forever) loop block. Inside the loop, there is a cyan 'もし' (If) block with a purple 'ボタン A ▼ が押されている' (Button A is pressed) condition. The 'なら' (Then) part of the 'もし' block contains a red '音を鳴らす' (Play sound) block. The '高さ (Hz)' (Pitch) field is empty, and the '傾斜 (°)' (Tilt) field is set to 'ローラ ▼' (Roller) with a '+' sign and a value of '90'. The '音量' (Volume) field is set to '× ▼' (Multiply) with a value of '5'. Below the 'もし' block is a cyan 'でなければ -' (Otherwise) block containing a red 'すべての音を停止' (Stop all sounds) block. The 'でなければ' block has a minus sign icon on its right side. At the bottom of the 'もし' block, there is a plus sign icon on the left side.

# タッチセンサ

- 右手でマイクロビットの「GND」ピンにさわった状態で、左手で「1」ピンにさわると「1」と表示されます
- 左手で「2」ピンにさわると「2」と表示されます
- 左手で「1」ピンと「2」ピンに同時にさわると「3」と表示されます

# タッチセンサ (プログラム)



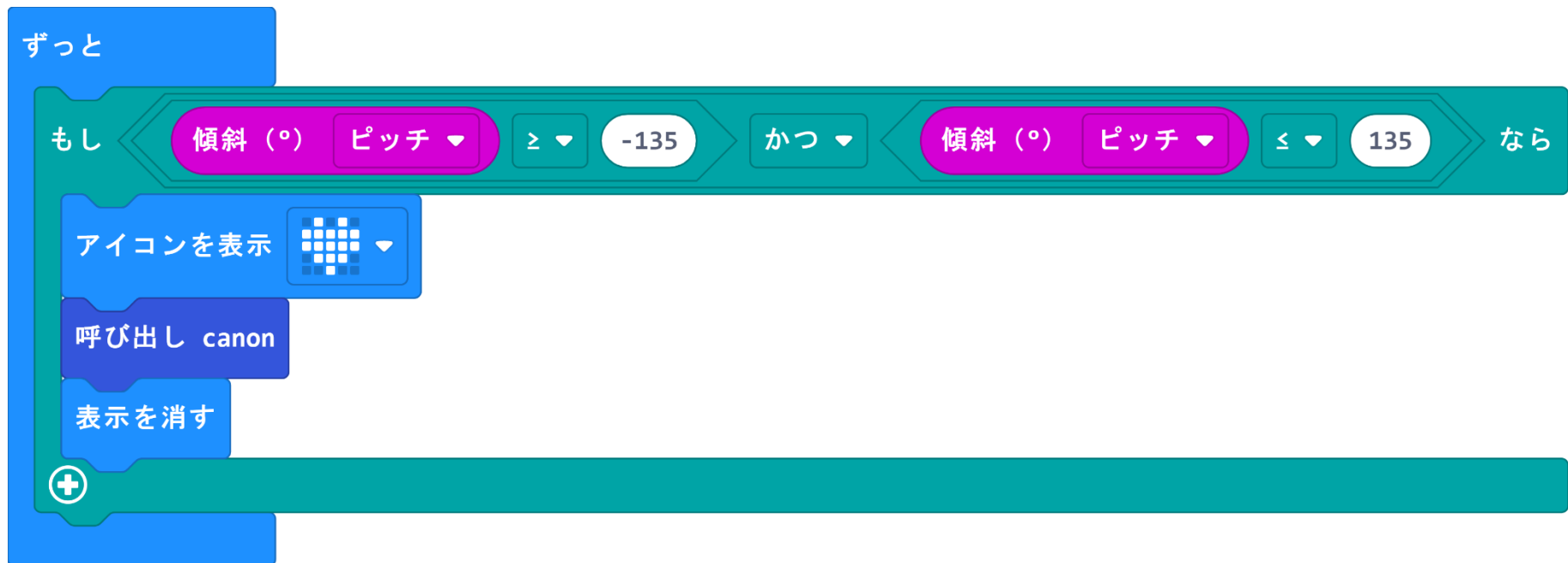
# オルゴール

- マイクロビットのかたむきを検知して、マイクロビットがおもてを向いたときに音楽がなります。同時にLEDでハートマークが表示されます
- マイクロビットをハコのふたにはりつけるとオルゴールになります

# オルゴール（説明）

- マイクロビットがおもてを向いて水平に置かれているとき、「傾斜 (°) ピッチ」は「0」になります
- 「傾斜 (°) ピッチ」が「-135」～「135」の時に音楽がなるようにします
- 音楽をならすための関数をつくり、その中に音符を入力していきます。好きなメロディーを入れてください（うしろのプログラム例では「カノン」という音楽を入れています）

# オルゴール (プログラム)



The image shows a Scratch script for an organ program. It starts with a 'ずっと' (Forever) loop block. Inside the loop, there is a 'もし' (If) block with two conditions: '傾斜 (°) ピッチ >= -135' and 'かつ' (and) '傾斜 (°) ピッチ <= 135'. If both conditions are met, the script performs three actions: 'アイコンを表示' (Show icon) with a grid icon, '呼び出し canon' (Call canon), and '表示を消す' (Hide). A plus sign (+) is visible at the bottom left of the loop block, indicating it can be expanded.

```
ずっと  
もし 傾斜 (°) ピッチ >= -135 かつ 傾斜 (°) ピッチ <= 135 なら  
  アイコンを表示 [Grid Icon]  
  呼び出し canon  
  表示を消す  
+
```



# メトロノーム

- マイクロビットから「ピッポッポッポッ」と音がなります
- 「B」ボタンをおすとテンポが早くなり、「A」ボタンをおすとおそくなります
- 「A」ボタンと「B」ボタンを同時におすと拍子が1ずつふえます（最大8まで）
- ゆさぶると拍子が1になります

# メトロノーム（説明1）

- 「拍子」「拍」「休符」という変数を用意し、最初に「拍子」を4、「拍」を0、「休符」を400にしておきます
- 最初に「拍子」の値を表示します
- 「A」ボタンと「B」ボタンを同時におしたとき、「拍子」が8より小さければ1だけ増やし、「拍子」の値を表示します
- ゆさぶられたとき、「拍子」を1にし、「拍子」の値を表示します

# メトロノーム（説明2）

- 「休符」の長さでテンポを調整します。「休符」の値が大きいほどテンポがおそく、小さいほどテンポが早くなります
- 「A」ボタンをおしたとき、「休符」が1000より小さければ20だけ増やします
- 「B」ボタンをおしたとき、「休符」が20より大きければ20だけへらします

# メトロノーム（説明3）

- ずっと1/16拍で音を鳴らし続けます
- 音を1回鳴らすたびに「休符」の時間だけ一時停止します
- 音を1回鳴らすたびに「拍」を1だけ増やします
- 鳴らす音は、「拍」が0のときは「真ん中のミ」、それ以外のときは「真ん中のド」にします
- もし「拍」が「拍子」以上になったら「拍」を0にします

# メトロノーム (プログラム)

最初だけ

変数 拍子 を 4 にする

変数 拍 を 0 にする

変数 休符 を 400 にする

数を表示 拍子

ボタン A+B が押されたとき

もし 拍子 < 8 なら

変数 拍子 を 1 だけ増やす

数を表示 拍子

ゆさぶられたとき

変数 拍子 を 1 にする

数を表示 拍子

ボタン A が押されたとき

もし 休符 < 1000 なら

変数 休符 を 20 だけ増やす

ボタン B が押されたとき

もし 休符 > 20 なら

変数 休符 を -20 だけ増やす

ずっと

もし 拍 = 0 なら

音を鳴らす 高さ (Hz) 真ん中のミ 長さ 1/16 拍

でなければ

音を鳴らす 高さ (Hz) 真ん中のド 長さ 1/16 拍

一時停止 (ミリ秒) 休符

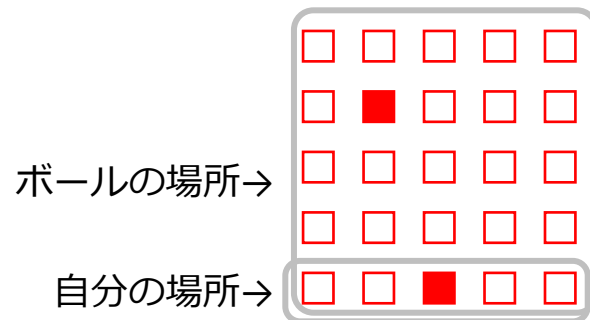
変数 拍 を 1 だけ増やす

もし 拍 ≥ 拍子 なら

変数 拍 を 0 にする

# シーソーゲーム

- LEDディスプレイのうち、一番下の1行は自分の場所を、全体がボールの場所を表します
- [A]ボタンをおすと、カウントダウンの後、ゲームがスタートします
- マイクロビットをかたむける（横方向）と、自分が左右に動きます
- ボールは上からまっすぐに落ちてくるので、自分に当たらないようににげます
- ボールが下に来たとき、自分に当たらなければ、得点が1点ふえ、次のボールが落ちてきます。またボールのスピードがはやくなります
- ボールが自分に当たると、ゲーム終了です



# シーソーゲーム（説明）

- 「高度なブロック」に「ゲーム」というブロックのグループがあるので、それを使います
- 「ボール」「自分」というふたつのスプライトを準備します。スプライトというのは、ゲームの中で動かすものです
- 「自分」スプライトは、マイクロビットのかたむきによって左右にうごきます
- 「ボール」スプライトは上から下にうごきます。横方向の場所はランダムにきまります
- 「ボール」と「自分」がぶつかったらゲーム終了です
- こまかい説明はむずかしいので、プログラム例を見て、仕組みを考えてみてください

# シーソーゲーム (プログラム)

```
最初だけ
  変数 ボール を スプライトを作成 x: 0 y: 0 にする
  変数 自分 を スプライトを作成 x: 2 y: 4 にする
  一時停止
  表示を消す

ボタン A が押されたとき
  呼び出し 前処理
  呼び出し ゲーム

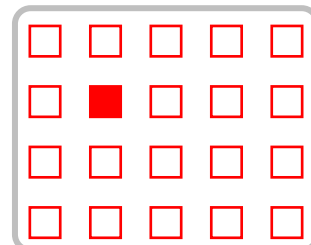
関数 前処理
  ボールの x に ランダムな数字を選択: 0 から 4 まで を設定する
  ボールの y に 0 を設定する
  点数を 0 にする
  変数 カウント を 3 にする
  くりかえし 3 回
    数を表示 カウント
    変数 カウント を -1 だけ増やす
  一時停止 (ミリ秒) 500
  表示を消す
  再開する
```

```
関数 ゲーム
  変数 スピード を 1000 にする
  もし 自分が他のスプライト ボール にさわっている ではない ならくりかえし
  もし 傾斜 (°) ロール < 0 なら
    自分の x を -1 だけ増やす
  でなければ
    自分の x を 1 だけ増やす
  もし ボールの y = 4 なら
    点数を 1 だけ増やす
    ボールの x に ランダムな数字を選択: 0 から 4 まで を設定する
    ボールの y に 0 を設定する
  もし スピード > 100 なら
    変数 スピード を -100 だけ増やす
  でなければ
    ボールの y を 1 だけ増やす
  一時停止 (ミリ秒) スピード
  ゲームオーバー
```

# ピンポンゲーム

- LEDディスプレイのうち、一番下の1行はラケットの場所を、その他の4行はボールの場所を表示します
- [A] または [B] ボタンをおすとラケットが左右に動きます
- [A] [B] ボタンを同時におすと、ゲームがスタートします
- ボールはまっすぐ、またはナナメに動き、壁や天井にぶつかるとはね返ります。はね返る方向は分かりません
- ボールが下に来たとき、ラケットに当たると「ピコーン！」と音がなり、得点が1点ふえます。またボールのスピードが少しはやくなります
- ボールがラケットに当たらなければ、ゲーム終了です
- リセットボタンをおすと、ゲームを再開できます

ボールの場所→



ラケットの場所→



# ピンポンゲーム (プログラム)

最初だけ

- 変数 ラケットx を 2 にする
- 点灯 x ラケットx y 4

ボタン A が押されたとき

- もし ラケットx > 0 なら
- 消灯 x ラケットx y 4
- 変数 ラケットx を -1 だけ増やす
- 点灯 x ラケットx y 4

ボタン B が押されたとき

- もし ラケットx < 4 なら
- 消灯 x ラケットx y 4
- 変数 ラケットx を 1 だけ増やす
- 点灯 x ラケットx y 4

ボタン A+B が押されたとき

- 変数 得点 を 0 にする
- 変数 時間 を 500 にする
- 変数 ボールx を ラケットx にする
- 変数 ボールy を 3 にする
- 変数 ボールdx を ランダムな数字を選択: -1 から 1 まで にする
- 変数 ボールdy を -1 にする
- 点灯 x ボールx y ボールy
- 一時停止 (ミリ秒) 時間
- 変数 ゲーム中 を 真 にする
- もし ゲーム中 ならくりかえし
- 呼び出し ボールを動かす
- もし ボールy = 3 なら
- 呼び出し 判定
- 点数を 得点 にする
- ゲームオーバー

# ピンポンゲーム (プログラム-関数)

```
関数 ボールを動かす
もし <ボールx> ≤ 0 なら
  変数 ボールdx を 1 にする
  変数 ボールdy を ランダムな数字を選択: -1 から 1 まで にする
+
もし <ボールx> ≥ 4 なら
  変数 ボールdx を -1 にする
  変数 ボールdy を ランダムな数字を選択: -1 から 1 まで にする
+
もし <ボールy> ≤ 0 なら
  変数 ボールdx を ランダムな数字を選択: -1 から 1 まで にする
  変数 ボールdy を 1 にする
+
もし <ボールy> ≥ 3 なら
  変数 ボールdx を ランダムな数字を選択: -1 から 1 まで にする
  変数 ボールdy を -1 にする
+
消灯 x ボールx y ボールy
変数 ボールx を ボールdx だけ増やす
変数 ボールy を ボールdy だけ増やす
点灯 x ボールx y ボールy
一時停止 (ミリ秒) 時間
```

```
関数 判定
もし <ボールx> = ラケットx なら
  メロディを開始する ビコーン! くり返し 一度だけ
  変数 得点 を 1 だけ増やす
もし <時間> > 100 なら
  変数 時間 を -10 だけ増やす
+
でなければ
  変数 ゲーム中 を 偽 にする
```