メディアアート・プログラミング I
第10回: 最終! - Patatapを作る!

2020年7月17日

東京藝術大学芸術情報センター（AMC）
田所淳

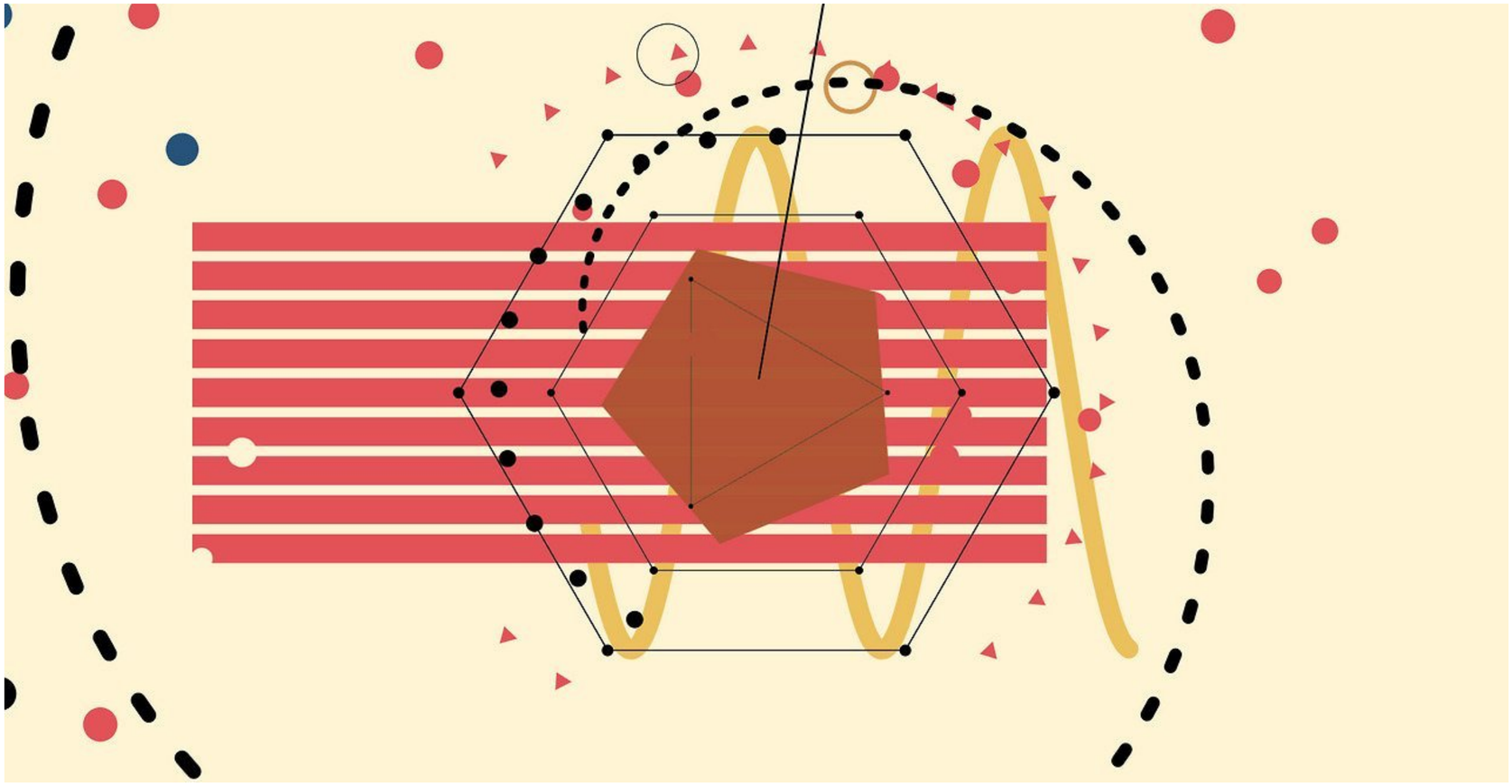
今日の内容

- 今まで学んできた内容を総括して、最終課題を制作
  - 形と色
  - アニメーション
  - くりかえし、配列
  - インタラクション
  - サウンド
  - クラス、オブジェクト

Patatapをつくる!

Patatapをつくる!

- Patatap - webブラウザで、音と映像をインタラクティブに操作する作品
- http://www.patatap.com/



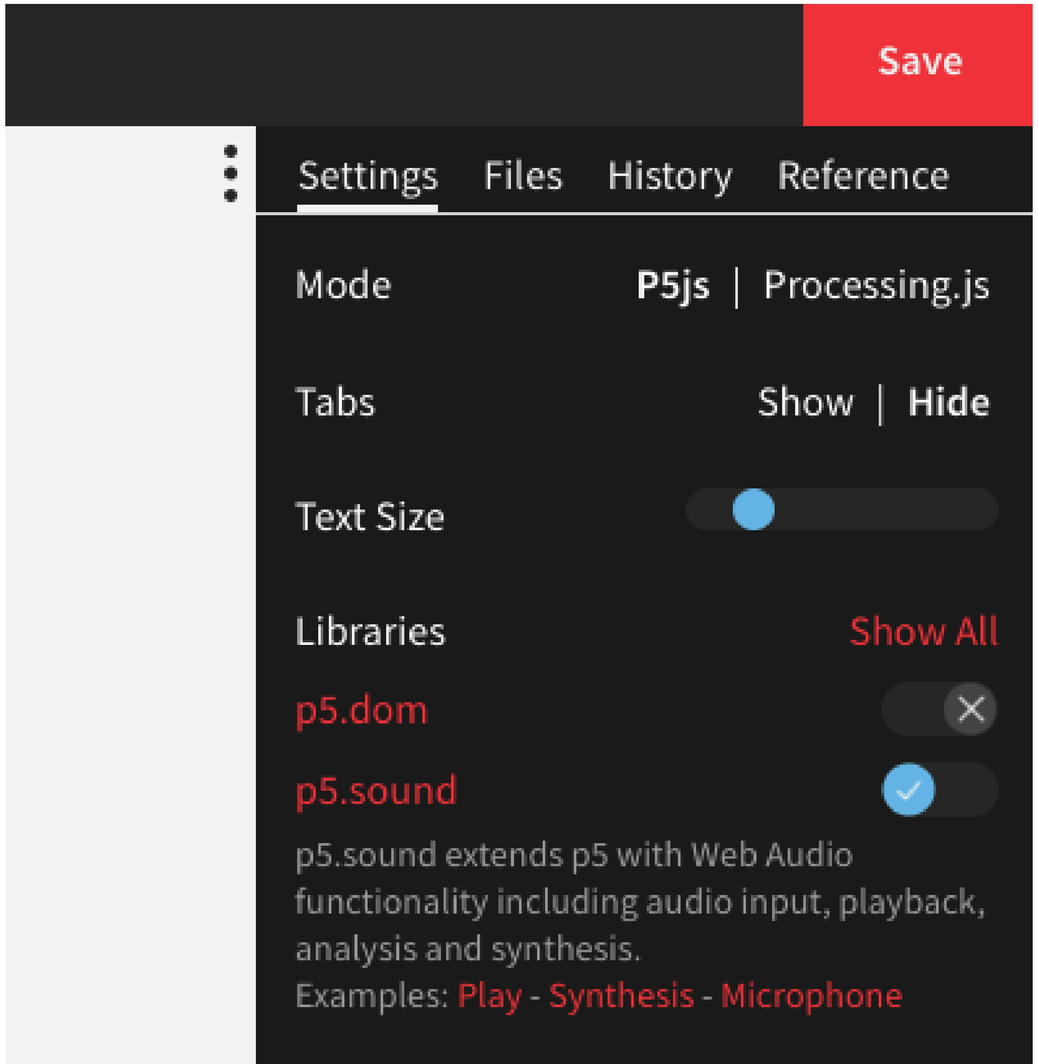
Patatapをつくる!

- PatatapをP5.jsで再現してみようと思います!

環境設定・サウンドファイルのアップロード

環境設定・サウンドファイルのアップロード

- SketchのsettingsでP5.soundをONにする



環境設定・サウンドファイルのアップロード

- 今日はあらかじめサウンドファイルをアップロードしたテンプレートを用意
- 下記のスケッチをForkして使用してください!

https://www.openprocessing.org/sketch/932615

サウンドの再生

サウンドの再生

- まずはシンプルにサウンドを再生してみる

```
let sample = [];

function preload() {
  sample[0] = loadSound('./se0.wav');
  sample[1] = loadSound('./se1.wav');
  sample[2] = loadSound('./se2.wav');
  sample[3] = loadSound('./se3.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  sample[0].play();
}

function draw(){
  background(0);
}
```

サウンドの再生

- サンプルの配列の番号を変えると、再生されるサウンドが変化するはず
- 一通り聴いてみる

インタラクティブにサウンドを再生 - ランダムに選択

インタラクティブにサウンドを再生

- キータイプで音が鳴るようにしてみる
- タイプするたびにランダムに音が選択されるように工夫してみる

インタラクティブにサウンドを再生

- sketch.js

```
let sample = [];

function preload() {
  sample[0] = loadSound('./se01.wav');
  sample[1] = loadSound('./se1.wav');
  sample[2] = loadSound('./se2.wav');
  sample[3] = loadSound('./se3.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw(){
  background(0);
}

function keyTyped(){
  let n = int(random(4));
  sample[n].play();
}
```

インタラクティブにサウンドを再生

- キータイプしてみる - ランダムに音が選択されて鳴るはず
- 連射しても大丈夫（音が途切れない）

キーボードに音を対応させる

キーボードに音を対応させる

- ランダムではなく、キーボードと音を一対一で対応させる
- 音が4つあるので、とりあえず a, s, d, f のキーで

キーボードに音を対応させる

- sketch.js

```
let sample = [];

function preload() {
  sample[0] = loadSound('./se0.wav');
  sample[1] = loadSound('./se1.wav');
  sample[2] = loadSound('./se2.wav');
  sample[3] = loadSound('./se3.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw(){
  background(0);
}

function keyTyped(){
  if (key == 'a') {
    sample[0].play();
  } else if (key == 's') {
    sample[1].play();
  } else if (key == 'd') {
    sample[2].play();
  } else if (key == 'f') {
    sample[3].play();
  }
}
```

キーボードに音を対応させる

- a, s, d, f それぞれのキーに音が対応したはず
- もちろん音を増やせば、もっとキーを増やせる

キーとビジュアルを対応させる

キーとビジュアルを対応させる

- 音だけでなく視覚的にもキーで変化するようにしてみる
- まずは単純に背景色を変えてみる

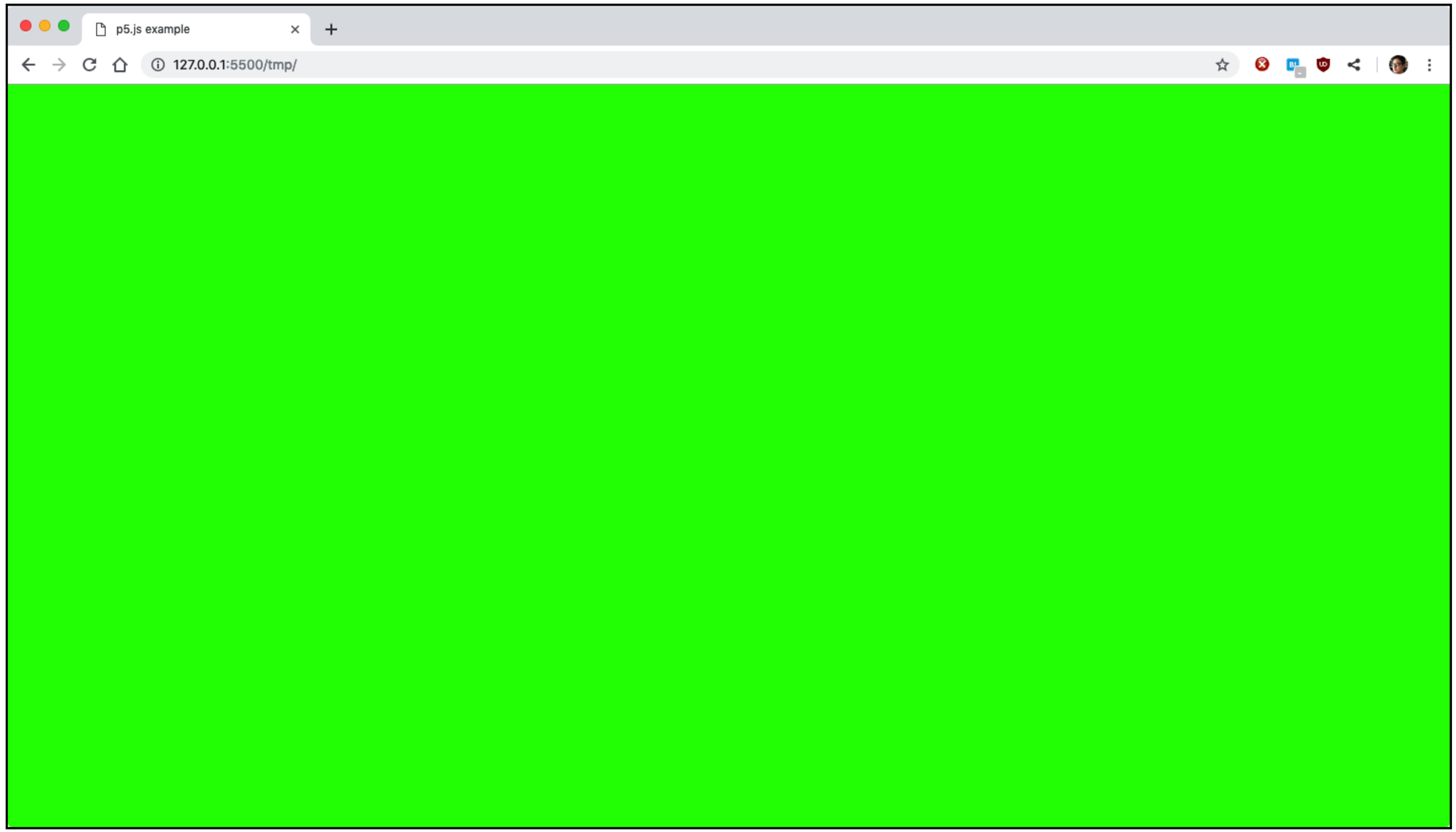
キーとビジュアルを対応させる

- sketch.js

```
let sample = [];
let bgColor;

function preload() {
  sample[0] = loadSound('./se01.wav');
  sample[1] = loadSound('./se02.wav');
  sample[2] = loadSound('./se03.wav');
  sample[3] = loadSound('./se04.wav');
  sample[4] = loadSound('./se05.wav');
  sample[5] = loadSound('./se06.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  bgcolor = color(0, 0, 0);
}

function draw(){
  background(bgcolor);
}

function keyTyped(){
  if (key == 'a') {
    sample[0].play();
    bgcolor = color(0, 100, 100);
  } else if (key == 's') {
    sample[1].play();
    bgcolor = color(60, 100, 100);
  } else if (key == 'd') {
    sample[2].play();
    bgcolor = color(120, 100, 100);
  } else if (key == 'f') {
    sample[3].play();
    bgcolor = color(180, 100, 100);
  } else if (key == 'g') {
    sample[4].play();
    bgcolor = color(240, 100, 100);
  } else if (key == 'h') {
    sample[5].play();
    bgcolor = color(300, 100, 100);
  }
}
```

キーとビジュアルを対応させる

- キーを押すと変化する背景色



アニメーションを対応させる

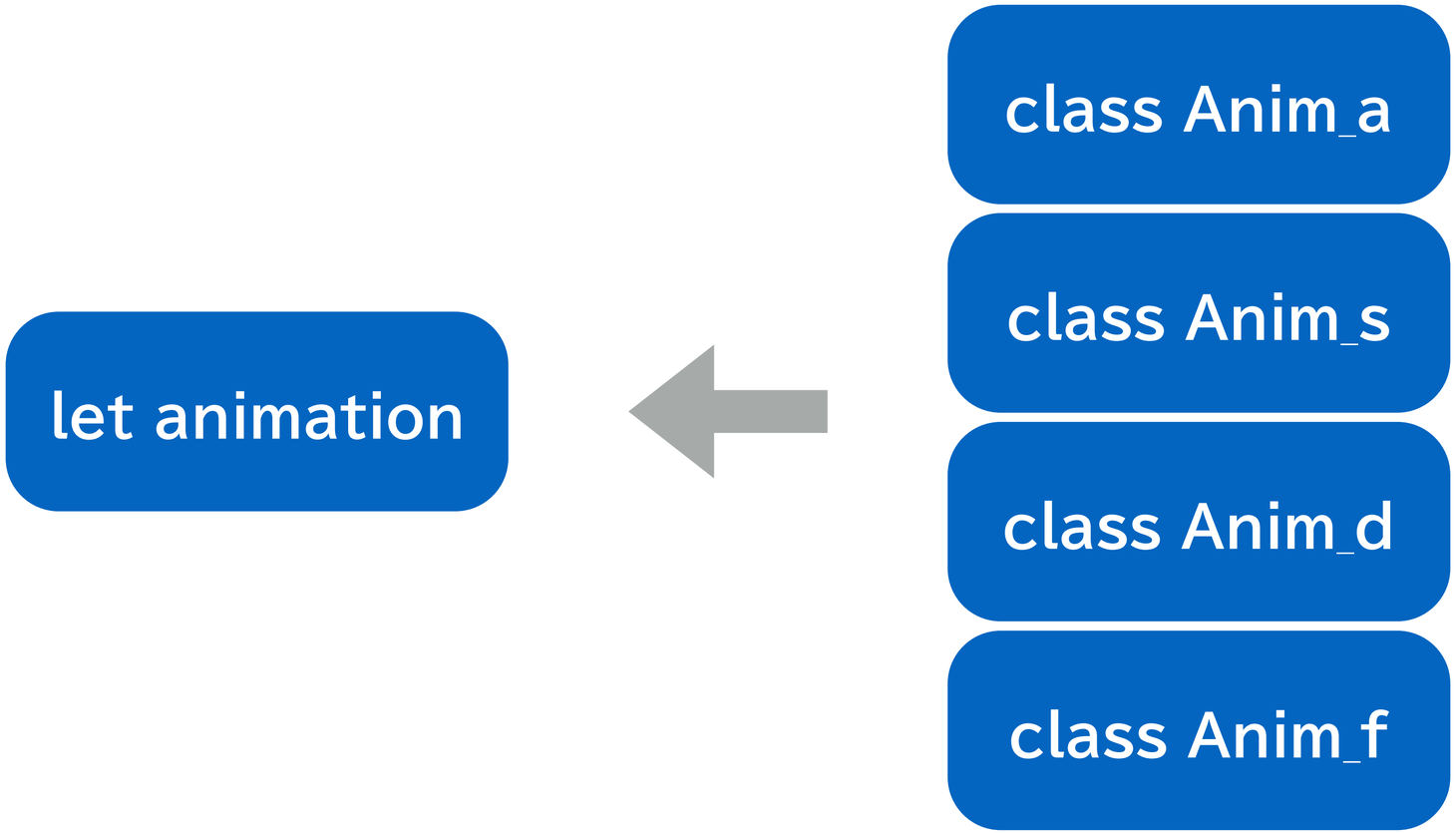
アニメーションを対応させる

- いよいよ、Patatapのようにキーにアニメーションを対応させてみます

- 背景色の時のように、keyTyped()の中にアニメーションを入れていく？
- どんどんコードが複雑になってしまいそう…
- もっと良い方法は無いか？

アニメーションを対応させる

- それぞれのアニメーションを、オブジェクトとして作る（classで実装）
- キーを押すたびに、オブジェクトを入れ替えるようにしてみる!



アニメーションを対応させる

- プログラムの大枠 - 1

```
let sample = [];
let animation;
let num;

function preload() {
  sample[0] = loadSound('./se0.wav');
  sample[1] = loadSound('./se1.wav');
  sample[2] = loadSound('./se2.wav');
  sample[3] = loadSound('./se3.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(0);
  if (animation) {
    animation.draw();
  }
}
```

```
function keyTyped() {
  if (key == 'a') {
    sample[0].play();
    animation = new Anim_a();
  } else if (key == 's') {
    sample[1].play();
    animation = new Anim_s();
  } else if (key == 'd') {
    sample[2].play();
    animation = new Anim_d();
  } else if (key == 'f') {
    sample[3].play();
    animation = new Anim_f();
  }
}
```

アニメーションを対応させる

- プログラムの大枠 - 2

```
// Animation A
class Anim_a {
  constructor() {
    //アニメーションAの初期化
  }
  draw() {
    //アニメーションAの描画
  }
}

// Animation S
class Anim_s {
  constructor() {
    //アニメーションSの初期化
  }
  draw() {
    //アニメーションSの描画
  }
}
```

```
// Animation D
class Anim_d {
  constructor() {
    //アニメーションDの初期化
  }
  draw() {
    //アニメーションDの描画
  }
}

// Animation F
class Anim_f {
  constructor() {
    //アニメーションFの初期化
  }
  draw() {
    //アニメーションFの描画
  }
}
```

アニメーションを対応させる

- それぞれのキーごとに簡単なアニメーションを作ってみる!
- あまり複雑でなくても面白い（はず）

アニメーションを対応させる

- アニメーション

```
// Animation A
class Anim_a {
  constructor() {
    this.x = width / 2;
    this.y = height / 2;
    this.diameter = 0;
    this.speed = 10;
  }
  draw() {
    noStroke();
    fill(0, 127, 255);
    ellipse(this.x, this.y, this.diameter,
this.diameter);
    this.diameter += this.speed;
  }
}

// Animation S
class Anim_s {
  constructor() {
    this.width = 0;
    this.speed = 80;
  }
  draw() {
    noStroke();
    fill(255, 0, 0);
    rectMode(CORNER);
    rect(0, 0, this.width, height);
    this.width += this.speed;
  }
}
```

アニメーションを対応させる

- アニメーション

```
// Animation D
class Anim_d {
  constructor() {
    this.rotate = 0;
    this.size = 0;
    this.speed = 50;
  }
  draw() {
    push();
    fill(255, 255, 0);
    noStroke();
    translate(width / 2, height / 2);
    rotate(radians(this.rotate));
    rectMode(CENTER);
    rect(0, 0, this.size, this.size);
    pop();
    this.rotate += this.speed;
    this.size += this.speed;
  }
}
```

```
// Animation F
class Anim_f {
  constructor() {
    this.bgColor = 255;
    this.speed = -2;
  }
  draw() {
    noStroke();
    fill(this.bgColor);
    rect(0, 0, width, height);
    this.bgColor += this.speed;
  }
}
```

アニメーションを対応させる

- 完成!!
- それぞれのキーに対応したアニメーションが再生されるはず
- さらにアニメーションを追加して、別のキーに割り振っていきましょう!

複数のアニメーションを重ねる

複数のアニメーションを重ねる

- Patatapをよく観察してみる
- 単体のアニメーションが再生されるのではなく、複数が折り重なっている
- 新たに追加されたアニメーションが上のレイヤーへ
- この仕組みを実装してみる!

複数のアニメーションを重ねる

- ポイント!
- アニメーションのオブジェクトを配列として表現する
- 指定した数よりレイヤーが増えたら先頭の配列を消去

複数のアニメーションを重ねる

- アニメーション

```
let sample = [];
let animation = [];
let num;
const maxAnim = 6;

function preload() {
  sample[0] = loadSound('./se01.wav');
  sample[1] = loadSound('./se02.wav');
  sample[2] = loadSound('./se03.wav');
  sample[3] = loadSound('./se04.wav');
  sample[4] = loadSound('./se05.wav');
  sample[5] = loadSound('./se06.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(0);
  if (animation.length > 0) {
    for (let i = 0; i < animation.length; i++) {
      animation[i].draw();
    }
  }
}
```

```
function keyTyped() {
  if (key == 'a') {
    sample[0].play();
    animation.push(new Anim_a());
  } else if (key == 's') {
    sample[1].play();
    animation.push(new Anim_s());
  } else if (key == 'd') {
    sample[2].play();
    animation.push(new Anim_d());
  } else if (key == 'f') {
    sample[3].play();
    animation.push(new Anim_f());
  } else if (key == 'g') {
    sample[3].play();
    animation.push(new Anim_g());
  } else if (key == 'h') {
    sample[3].play();
    animation.push(new Anim_h());
  }
  if(animation.length > maxAnim){
    animation.splice(1, 1);
  }
}
... (後略) ...
```

複数のアニメーションを重ねる

- 完成!!
- Patatapのようにアニメーションが折り重なって再生されるはず!

本日の課題&最終課題 Patatapを作る

本日の課題&最終課題 Patatapを作る

- 制作用のテンプレート作成しました
- 下記のスケッチをforkして使用してください

https://www.openprocessing.org/sketch/932636

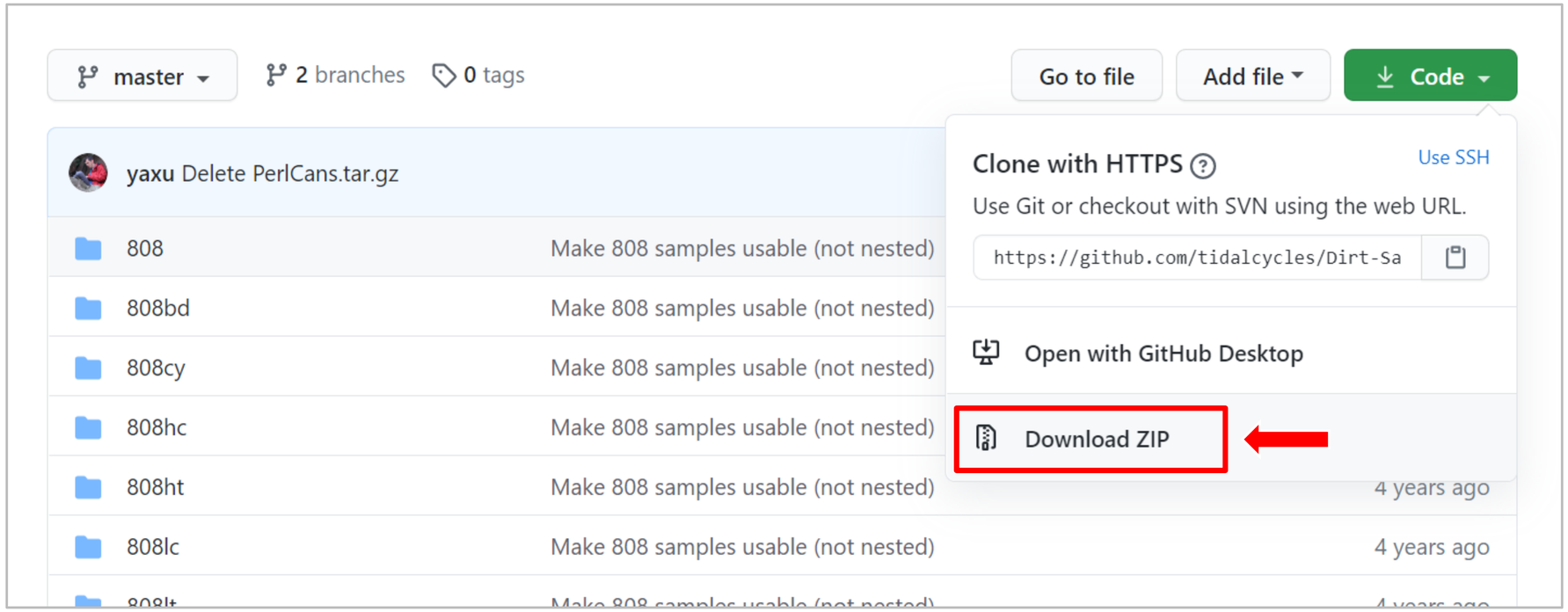
本日の課題&最終課題 Patatapを作る

- 音ファイルの入手先 1
  - FreeSound
  - https://freesound.org/

本日の課題&最終課題 Patatapを作る

- 音ファイルの入手先 2
  - Dirt Samples - TidalCycles用のサウンドサンプル集
  - https://github.com/tidalcycles/Dirt-Samples

  - ダウンロードは右上のCode → Download Zip を選択



本日の課題&最終課題 Patatapを作る

- アニメーション制作の考え方
  - アニメーションを切り替える部分は、テンプレートをそのまま使用
  - setup() draw() keyTyped() の部分

- 必ずしも、凝ったアニメーションを作る必要はない
  - シンプルなアニメーションもインタラクティブで変化することで面白くなる
  - 何か1つの要素が変化するだけでも
    - 色が変化
    - 大きさが変化
    - 角度が変化
    - 位置が変化
    - ...etc

  - 実際にコーディングしながら解説

課題の提出方法

本日の課題提出

- 本日中に出来たものを以下に提出してください
- 作品のOpenProcessingのURL(アドレス)を下記のフォームから投稿

https://bit.ly/3h6IiHe

課題の最終提出

- 今回は最後の課題なので、正式な提出は再来週までとします
- 作品のOpenProcessingのURL(アドレス)を下記のフォームから投稿

https://bit.ly/32px0cI

- 締切: 7月30日 (木)
- 再来週 (7月31日、13:00-) 自由参加(出席カウント外)で講評会を行います