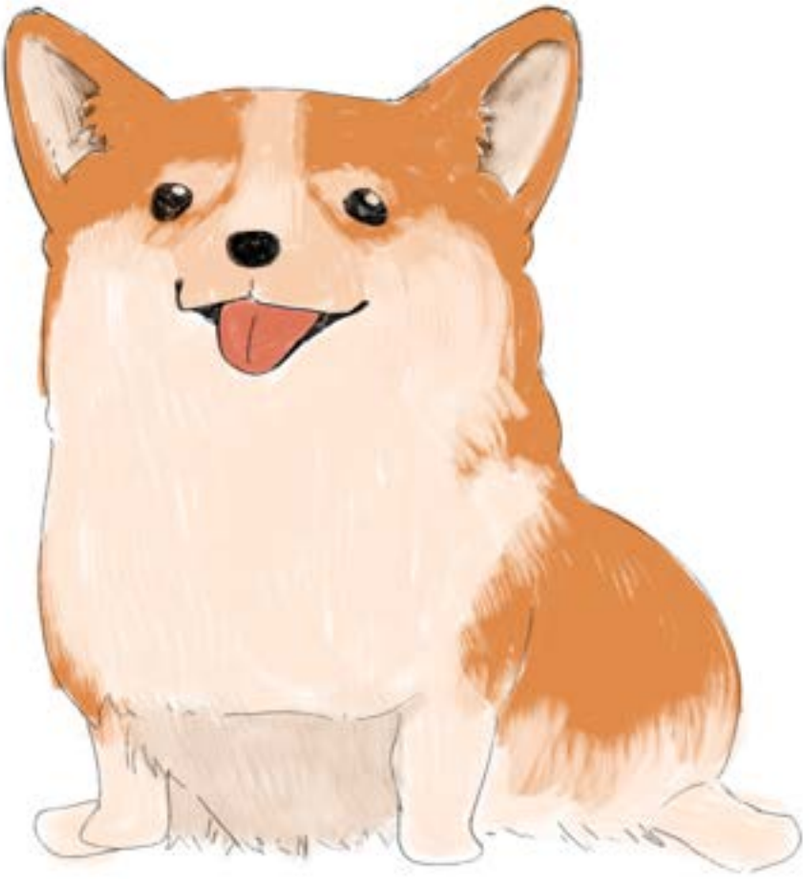
ゼミ展

制作過程

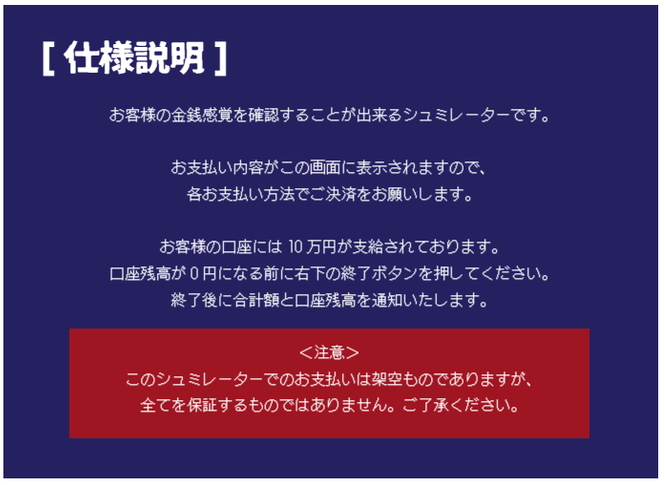


D12　日江井沙来

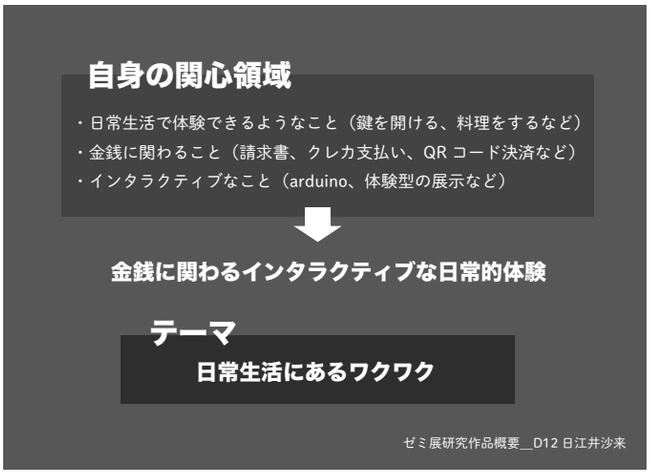
テーマ決め

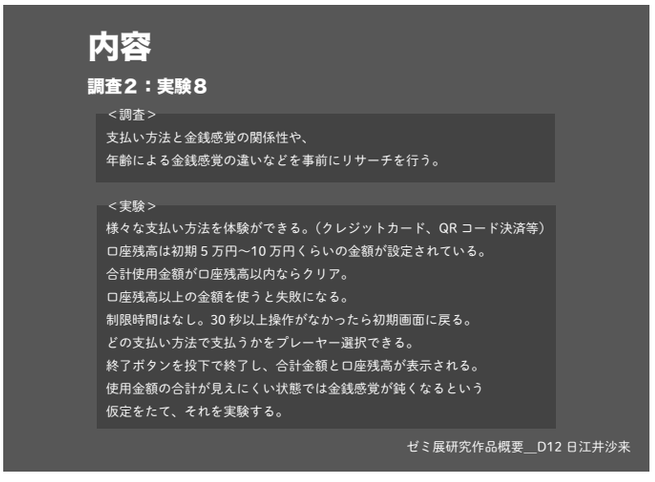
最初は金銭関連の何かインタラクティブなものを
作りたいという漠然としたテーマだけあったので、
そこから、デジタルで支払いをすると金銭感覚が
おかしくなるよね。というところから、
体験者の金銭感覚を確認してみよう！といった感じの
『金銭感覚シュミセーター』というのを考えてみた。
試作でビジュアルを作ってみたのだが、
よく考えたら実際に支払ってもないのに金銭感覚も
クソもないよな〜と思い、断念した。
その後デジタル決済そのものの面白さの方に着目して、
体験者（というか私）が面白いな。と思える
決済アプリを作ることにした。
いったん決済した時にかわいい動物を出したい。
と思い、試行錯誤したが
ある時キュートアグレッションって金銭の消費と感情が
似てるかも。と思い、現在の『きゅーあぐ PAY』の
制作に至った。

最初の『金銭感覚シュミレーター』のビジュアル



最初考えてた展示構想





砂時計に埋もれるねこ（没）

今の『きゅーあぐ PAY』の構想

テーマ：私が使ってみたい電子決済

使ってみてワクワクする＆お金を消費することを楽しくしてくれるもの

⇩

かわいい動物をいじめたくなる行為（キュートアグレッション）をもとに
かわいい動物が消費額によって可哀想な目にあっていく仕様。

展示名：きゅーあぐ PAY
実際の決済の画面をモニターに縦型で表示。
決済は aruduino を使って実際の QR コード決済の方法と同じようにスキャナーのようなものを使用している感じにする。

動物のバリエーションは３種類。ランダム表示される。
消費額は 1 日ごとに蓄積型にしたい。

⇩

試作の画面録画　https://drive.google.com/file/d/1eH02DF6hVFnmZfmDYlaqmJXIMws9z6Mn/view?usp=drive_link

画面画像

青とピンクのグラデーションを
基調とした画面。
これらは illustrator で
制作した。その他の文字は
processing 側に文字を
インポートして表示させている。

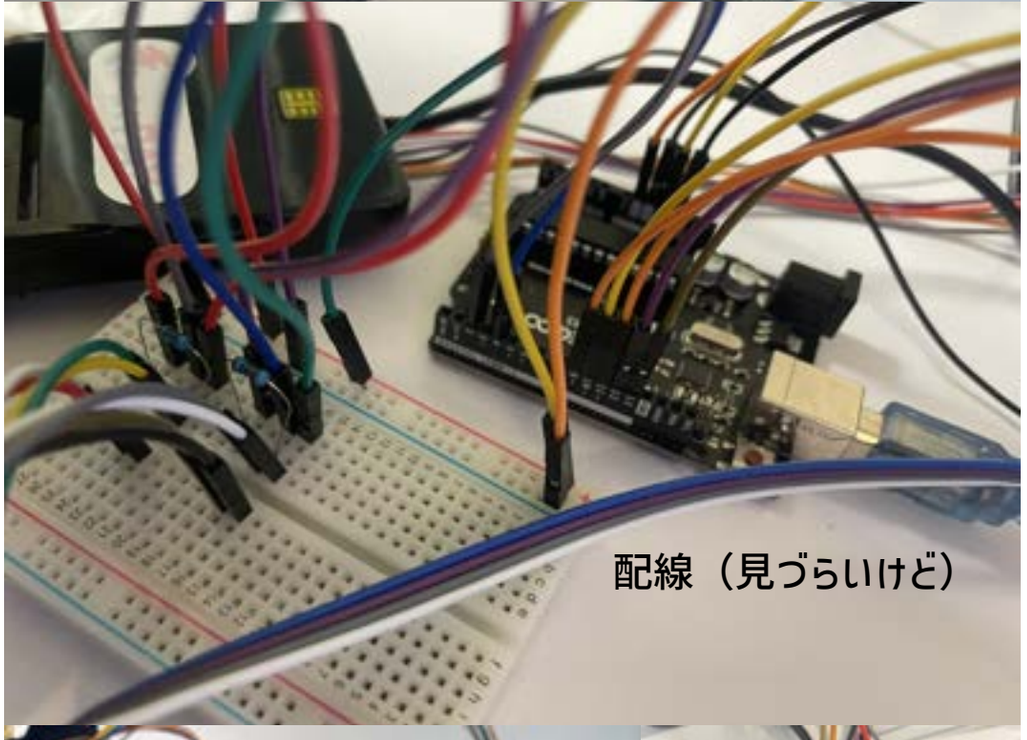
解像度がデカすぎて prcessing が
動かなくなることも多々あり…

キュートアグレッションについて

キュートアグレッションとは、赤ちゃんや動物など、
かわいいものを見たときに感じるぎゅっとしたり、
軽く噛んだりしたくなるような衝動のこと。
これは、人間の脳の報酬系と関連していて、
かわいいものを見ると、脳内でドーパミンが分泌され、
快感を感じる。この快感が過剰になると、
それを調節するために、キュートアグレッションのような
行動が起こると考えられているらしい。
キュートアグレッションは、多くの人が経験する正常な
反応で、特に共感能力が高い人や、世話好きな人に
多く見られる傾向があるらしい。
今回の作品では、実際に動物をいじめちゃっているが、
実際の場合直接危害を加えるケースは少ない。
しかし、普段いじめたくてもいじめられない動物たちを
いじめられることにもさらに快感を感じてもらいたいと
思っている。

配線とか


全体図


配線（見づらいけど）


ボタンたち


MFRC522

arduino の配線。
スイッチとカードリーダーを接続している。

スイッチは
タクトスイッチを使い、
pin2,4 に接続している。
台は百均の箱をはんだごてで無理やり穴を開けた。
段差があったので、それは粘土で埋め合わせてその上にきらきらシールを貼った。
何回も接触不良になり、グルーガンでガチガチにしたあと根元側で両方とも接触不良になった時ははんだ付けからやり直しで萎えてしまった…

カードリーダーは、MFRC522というモジュールを使っている。
カードリーダーを買ったのだが、パソコンに接続できなくて泣く泣く MFRC522 を使うことに。これも何度も壊れて何度も買い直してやっと接続できた時は嬉しすぎて飛び回った！

小物系

クレジットカード、QR コード画面の
デザインは全て illustrator で制作した。
クレジットカードは、写真紙を貼ると厚さで
カードリーダーに差し込めなくなてしまうので、
普通紙にレジン液でコーティングして
厚さを抑えた。カード自体も若干削った。
QR コードリーダーは買ったのだが、
普通に使えたのでそのまま使用。
QR コード画面は色々な決済アプリの
決済画面を参考にした。
QR コードの色が薄いのが懸念点だったが、
実際に読み込めることを確認した。安心。
バーコードリーダーは、実際に体験では
使わないのだが、なんか画面の横にあったら
決済している感が増すかな。と思い、
買ってしまった。
展示するまでどうなるかはわからない…

コード全文（arduino）

```
#include <SPI.h>
#include <MFRC522.h>


#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance.

const int button1Pin = 2; // ボタン 1 のピン
const int button2Pin = 4; // ボタン 2 のピン

int lastButton1State = HIGH;   // ボタン 1 の前回の状態
int lastButton2State = HIGH;   // ボタン 2 の前回の状態

// カード 1 の UID
byte card1UID[] = {0x53, 0xDA, 0x56, 0x25}; // 実際のカード UID に置き換える

// カード 2 の UID
byte card2UID[] = {0x03, 0x50, 0x71, 0x28}; // 実際のカード UID に置き換える

// 2 つの UID を比較する関数
boolean compareUIDs(byte *uid1, byte *uid2) {
  for (byte i = 0; i < 4; i++) {
    if (uid1[i] != uid2[i]) {
      return false;
    }
  }
  return true;
}

void setup() {
  Serial.begin(9600);     // Initialize serial communications with the PC
  while (!Serial);        // Do nothing if no serial port is opened (added for Arduinos
based on ATmega32U4)
  SPI.begin();            // Init SPI bus
  mfrc522.PCD_Init();    // Init MFRC522 card
  pinMode(button1Pin, INPUT_PULLUP);
  pinMode(button2Pin, INPUT_PULLUP);
}

void loop() {
  // ボタン 1 の状態を読み取る
  int reading1 = digitalRead(button1Pin);
  if (reading1 != lastButton1State) { // 状態が変化した場合のみ
    if (reading1 == LOW) {            // ボタンが押されたとき
      Serial.println("Button 1 pressed");
    } else {                          // ボタンが離されたとき
      Serial.println("Button 1 released");
    }
    lastButton1State = reading1; // 現在の状態を保存
  }
  // ボタン 2 の状態を読み取る
  int reading2 = digitalRead(button2Pin);
  if (reading2 != lastButton2State) { // 状態が変化した場合のみ
    if (reading2 == LOW) {            // ボタンが押されたとき
      Serial.println("Button 2 pressed");
    } else {                          // ボタンが離されたとき
      Serial.println("Button 2 released");
    }
    lastButton2State = reading2; // 現在の状態を保存
  }

  // カードのチェック
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  // カードの UID と一致するか判定
  if (compareUIDs(mfrc522.uid.uidByte, card1UID)) {
    Serial.println("Card 1 detected");
  } else if (compareUIDs(mfrc522.uid.uidByte, card2UID)) {
    Serial.println("Card 2 detected");
  }

  // Halt PICC
  mfrc522.PICC_HaltA();
  // Stop encryption on PCD
  mfrc522.PCD_StopCrypto1();
}
```

ボタンとカードリーダーの接続用のコード。
ボタンが押された！、カードを認識した！
という信号を processing 側に送って、
その信号を利用して processing 側で
制御している。

カードはそれぞれ認識コードがあるので、
それをまた別のコードで確認する。
めんどくさかった。

コード全文（processing）

```
import processing.video.*;
import processing.serial.*;
import ddf.minim.*;
import processing.sound.*;

PFont font1, font2;

Serial myPort;  // シリアルポート

Minim minim;
SoundFile[] sounds; // AudioPlayer の配列を宣言
SoundFile[] effects;
AudioPlayer cardDetectedSound; // 効果音用の AudioPlayer
boolean soundPlayed = false; // 効果音を再生したかどうか
boolean soundPlayedInCase4 = false; // case 4 内でサウンドが再生されたかを追跡する
boolean effectPlayedInCase5 = false; // case 5 での再生を制御するフラグ
boolean videoLoaded = false; // 動画がロードされているかを示すフラグ

PImage img1, img2, img3, img4; // 画像
Movie video1;  // 動画
Movie[] videos; // Movie 型の配列を宣言
PImage[] case3Images; // case 3 用の画像を格納する配列
PImage[] case4Images; // case 4 用の画像を格納する配列
PImage[] case5Images; // case 5 用の画像を格納する配列

int enterPressCount = 0;  // エンターキーのカウント
boolean overlayVisible = false;    // 画像の表示フラグ
boolean animationActive = false;    // アニメーションフラグ
boolean videoPlaying = false;    // video の再生フラグ
boolean video1Finished = false;    // video1 終了フラグ
boolean delayStarted = false;    // video ディレイフラグ
boolean returnToStart = false;    // 最初の画面に戻るフラグ

float aspectRatio = 9.0 / 16.0; // 縦横比（9:16）

int videoDelayTime = 2000;    // video ディレイ時間
int videoStartTime;    // video 開始時間
float fadeAlpha = 0;    // 不透明度

int randomAmount = 0;    // ランダム金額 ( 数値に変更
int totalAmount = 0;    // 合計金額
int remainingCount = 0; // 残りの回数

int loadingAngle = 0;    // ローディング用角度

int GamenYoko = 1080;
int GamenTate = 1920;


// 各 case の滞在時間 ( ミリ秒 )
int case1DisplayTime = 120000;
int case3DisplayTime = 5000;
int case4DisplayTime = 120000;
int case5DisplayTime = 5000;

int caseStartTime; // 各 case に入った時間を記録する変数
int randomIndex; // ランダムなインデックスを格納する変数 ( グローバル変数 )
ImageTextPair pair; // ランダムに選択したペアを格納する変数 ( グローバル変数 )
int videoButtonRemainingCount;


// 画像とテキスト、音声のペアを格納するクラス
class ImageTextPair {
  PImage image;
  String[] texts;
  AudioPlayer sound; // AudioPlayer を追加

  ImageTextPair(PImage image, String[] texts) { // コンストラクタに sound を追加
    this.image = image;
    this.texts = texts;
    this.sound = sound;
  }
}

// 画像とテキストのペアの配列
ImageTextPair[] pairs; // クラス定義の外で宣言

// randomAmount を生成する関数
void generateRandomAmount() {
  randomAmount = (int) random(500, 2001); // ランダム金額を生成 ( 整数 )
  totalAmount += randomAmount;              // 合計金額に追加
  videoButtonRemainingCount = randomAmount / 100; // ボタンの残り回数を計算
}

void movieEvent(Movie m) {
  m.read();
}


void settings() {
  img1 = loadImage(" 画面 _01.png");
  img2 = loadImage(" 画面 _02.png");
  img3 = loadImage(" 画面 _03.png");
  img4 = loadImage(" 画面 _04.png");

  //fullScreen(0);

  size(GamenYoko, GamenTate); // size() はデフォルトの向きで呼び出す
  smooth();

}
```

```
void setup() {

  //// 描画範囲を計算
  //int windowWidth = displayWidth;
  //int windowHeight = (int)(displayWidth / aspectRatio);
  // surface.setSize(windowWidth, windowHeight);
  //println("Width: " + windowWidth + ", Height: " + windowHeight);



  // シリアルポートを開く
  String portName = "/dev/cu.usbmodem101"; // 使用するシリアルポートを選択
  myPort = new Serial(this, portName, 9600);
  video1 = new Movie(this, "check.mp4");
  minim = new Minim(this);
  sounds = new SoundFile[3]; // AudioPlayer の配列を初期化
  effects = new SoundFile[3];

  sounds[0] = new SoundFile(this, " やめて〜 (mp3cut.net).wav"); // サモエドの音声ファイルをロード
  sounds[1] = new SoundFile(this, " 黒田さん (mp3cut.net).wav"); // ポメラニアンの音声ファイルをロード
  sounds[2] = new SoundFile(this, " 伊藤さん (mp3cut.net).wav"); // コーギーの音声ファイルをロード

  // サウンドファイルをロード
  effects[0] = new SoundFile(this, " ペタッ .wav");
  effects[1] = new SoundFile(this, " しょげる .wav");
  effects[2] = new SoundFile(this, " 散歩日和 _2 (mp3cut.net).wav");



// プリロードしておく（全てのサウンドで）
  for (SoundFile sound : sounds) {
    sound.play();
    sound.stop();
  }
  for (SoundFile effect : effects) {
    effect.play();
    effect.stop();
  }

    // effects[2] をループ再生開始
  effects[2].loop(); // 散歩日和 _2.wav をループ再生

  cardDetectedSound = minim.loadFile("koukaon (mp3cut.net).wav"); // 効果音をロード

  font1 = createFont("Corporate-Logo-Medium.otf", 48); // フォント指定
  font2 = createFont("Corporate-Logo-Bold.otf", 48); // フォント指定

  // 画像とテキスト、音声のペアを初期化
  pairs = new ImageTextPair[3]; // 配列のサイズを 3 に設定
  pairs[0] = new ImageTextPair(
    loadImage("samoedo_01.png"),
    new String[] {
      " サモエドを ",
      " 口をビヨンビヨンに伸ばせます "
    }
  );

  pairs[1] = new ImageTextPair(
    loadImage("pome_01.png"),
    new String[] {
      " ポメラニアンを ",
      " 頭をぺしゃぺしゃにおしつぶせます "
    }
  );

  pairs[2] = new ImageTextPair(
    loadImage("corgi_01.png"),
    new String[] {
      " コーギを ",
      " おしりを食パンにできます "
    }
  );

  // case 3 用の画像を配列に格納
  case3Images = new PImage[3];
  case3Images[0] = loadImage("samoedo_01.png");
  case3Images[1] = loadImage("pome_01.png");
  case3Images[2] = loadImage("corgi_01.png");

    // case 4 用の画像を配列に格納
  case4Images = new PImage[3];
  case4Images[0] = loadImage("samoedo_03.png");
  case4Images[1] = loadImage("pome_03.png");
  case4Images[2] = loadImage("corgi_03.png");

  // case 5 用の画像を配列に格納
  case5Images = new PImage[3];
  case5Images[0] = loadImage("samoedo_02.png");
  case5Images[1] = loadImage("pome_02.png");
  case5Images[2] = loadImage("corgi_02.png");

  videos = new Movie[3]; // videos 配列を初期化
  videos[0] = new Movie(this, "samoedo.mp4");
  videos[1] = new Movie(this, "pome.mp4");
  videos[2] = new Movie(this, "corgi.mp4");

    // 動画の再生準備を開始
  for (int i = 0; i < videos.length; i++) {
    videos[i].loop(); // 動画をループ再生して読み込みを開始
    videos[i].pause(); // 一時停止してロードだけ行う
  }
}



void draw() {
  //scale(0.5);

  background(255);
  if (returnToStart) {
    // 最初の画面に戻る（合計金額はそのまま）
    enterPressCount = 0;
    returnToStart = false;
    randomAmount = 0;

    // case 2 の再試行を可能にするためのフラグのリセット
    video1Finished = false;
    videoPlaying = false;
```

```
overlayVisible = false;
    animationActive = false;
    fadeAlpha = 0;
    delayStarted = false;
    loadingAngle = 0;
    caseStartTime = 0; // caseStartTime をリセット
    randomIndex = (int) random(pairs.length); // randomIndex を再生成

    // video1 のリセット
    video1.stop();
    video1.jump(0);
  }



 // シリアルポートからデータを読み取る
while (myPort.available() > 0) {
  String inBuffer = myPort.readStringUntil('\n');
  if (inBuffer != null) {
    inBuffer = trim(inBuffer); // 不要な空白を削除

    // ボタン 1 が押された場合 (case0 → case1)
    if (inBuffer.equals("Button 1 pressed") && enterPressCount == 0) {
      enterPressCount = 1; // case 1 へ移行
      caseStartTime = 0; // caseStartTime をリセット
      effects[0].jump(0); // サウンドを再生
      effects[0].play();
    }

    // カードが読み取られた場合 (case1 → case2)
    else if ((inBuffer.equals("Card 1 detected") || inBuffer.equals("Card 2
detected")) && enterPressCount == 1) {
      enterPressCount = 2; // case 2 へ移行
      caseStartTime = 0; // caseStartTime をリセット
      overlayVisible = true; // カード認識時にも overlayVisible を true に設定
    }

    // ボタン 2 が押された場合 (case4 で動画再生 )
    if (inBuffer.equals("Button 2 pressed") && enterPressCount == 4) {
      videoButtonRemainingCount--; // ボタンの残り回数を減らす

      // 動画を停止し、再生位置をリセットして再生し直す
      videos[randomIndex].stop();  // 再生中の場合は停止
      videos[randomIndex].jump(0); // 動画を最初に戻す
      videos[randomIndex].play();  // 再生開始

      // 必要ならサウンド再生
      sounds[randomIndex].jump(0);
      sounds[randomIndex].play();

      // case 4 で動画を再生するためのフラグを設定
      soundPlayedInCase4 = true;
    }
  }
}
```

```
switch (enterPressCount) {
case 0:
  // エンター 1 回目：1 枚目の画像を表示
  image(img1, 0, 0, GamenYoko, GamenTate);
  break;



case 1:
  // case 1 に入った時間を記録
  if (caseStartTime == 0) {
    caseStartTime = millis();
  }
  // エンター 2 回目：2 枚目の画像を表示
  image(img2, 0, 0, GamenYoko, GamenTate);

  // 動作しないと case0 に戻る
  if (millis() - caseStartTime >= case1DisplayTime) {
    enterPressCount = 0;
    returnToStart = true; // リセット処理を実行
    caseStartTime = 0; // caseStartTime をリセット
  }
  break;



case 2:
  if (caseStartTime == 0) {
    caseStartTime = millis();
  }
  image(img4, 0, 0, GamenYoko, GamenTate);

  if (overlayVisible && !cardDetectedSound.isPlaying()) { // 再生中でない
場合のみ
    cardDetectedSound.rewind(); // 再生位置を先頭に戻す
    cardDetectedSound.play(); // 効果音を再生
  }

  // video1 の再生処理
  if (!video1Finished) {
    if (!video1.isPlaying())
      video1.play();
    if (video1.available())
      video1.read();

    image(video1,100, 900, 900, 900);

    // video1 終了の判定
    if (video1.time() >= video1.duration()) {
      video1.stop();
      video1Finished = true;
      delayStarted = false; // video2 の遅延表示を初期化
      animationActive = true; // アニメーションを開始
      enterPressCount = 3; // case 3 へ移行
      caseStartTime = 0; // caseStartTime をリセット
    }
  }
  break;
```

```
  case 3:
    // case 3 に入った時間を記録
    if (caseStartTime == 0) {
      caseStartTime = millis();
      generateRandomAmount(); // randomAmount を生成する関数を呼び出す
      randomIndex = (int) random(pairs.length); // ランダムなインデックスを生成
    }
    image(img3, 0, 0, GamenYoko, GamenTate);

    // ランダムに選択したペアの画像を表示
    pair = pairs[randomIndex]; // ランダムに選択したペアを取得
    image(case3Images[randomIndex], 120, 680, 850, 830); // case 3 用の画像を表示

    fadeAlpha = min(fadeAlpha + 10, 255);
    textSize(24);
    fill(100, fadeAlpha); // テキスト色
    textAlign(CENTER, CENTER); // 水平・垂直揃えをセンターに設定
    textFont(font1);
    textSize(50); // " 今回のかわいいは…" のフォントサイズを指定
    text(" 今回のかわいいは…", width / 2 - 150, 600);

    textFont(font2);
    textSize(70); // 動物名のフォントサイズを指定

    // randomIndex に応じて動物名を表示
    switch (randomIndex) {
    case 0:
      text(" サモエド ", width / 2 + 200, 595);
      break;
    case 1:
      text(" ポメラニアン ", width / 2 + 220, 595);
      break;
    case 2:
      text(" コーギー ", width / 2 + 200, 595);
      break;
    }

    textFont(font1);
    textSize(40);
    text(" 今回の決済額は ", width / 2 - 230, 1600);
    text(" なので…", width / 2 + 280, 1600);
    textFont(font2);
    textSize(70);
    text(randomAmount + " 円 ", width / 2 + 60, 1588);

    if (millis() - caseStartTime >= case3DisplayTime) {
      enterPressCount = 4;
      caseStartTime = 0; // caseStartTime をリセット
    }
    break;


  case 4:
    // case 4 に入った最初のフレームでのみ randomIndex を初期化
    if (caseStartTime == 0) {
      caseStartTime = millis();
      fadeAlpha = 0; // fadeAlpha を 0 に初期化
    }

    image(img3, 0, 0, width, height);
    image(case4Images[randomIndex], 65, 630, 950, 830); // case 4 用の画像を表示

   // ここで動画の再生を制御
   if (soundPlayedInCase4) {
     // soundPlayedInCase4 が true になった最初のフレームでのみ実行
       // 動画が最後まで再生されたらフラグをリセット
   if (videos[randomIndex].time() >= videos[randomIndex].duration() - 0.1) {  // 終了時間の 0.1 秒前になったら
     soundPlayedInCase4 = false;
     videos[randomIndex].pause(); // 動画を一時停止
     videos[randomIndex].jump(0); // 再生位置を 0 に戻す
   }

    fadeAlpha = min(fadeAlpha + 20, 255);
    pushStyle();
    tint(255, fadeAlpha);
    image(videos[randomIndex], 65, 630, 950, 830);
    popStyle();

    // 動画が最後まで再生されたらフラグをリセット
    if (videos[randomIndex].time() >= videos[randomIndex].duration()) {
      soundPlayedInCase4 = false;
      videos[randomIndex].stop(); // 動画を停止
    }
   }
    // テキストの表示
    fill(100, fadeAlpha); // テキスト色
    textAlign(CENTER, CENTER); // 水平・垂直揃えをセンターに設定
    textFont(font2);
    textSize(50);

    // randomIndex に応じてテキストを切り替え
    switch (randomIndex) {
    case 0:
      text(pairs[randomIndex].texts[0] + ( randomAmount / 100 ) + " 回 ", width / 2, 520); // 1 つ目のテキスト
      text(pairs[randomIndex].texts[1], width / 2, 590); // 2 つ目のテキスト
      break;
    case 1:
      text(pairs[randomIndex].texts[0] + ( randomAmount / 100 ) + " 回 ", width / 2, 520); // 1 つ目のテキスト
      text(pairs[randomIndex].texts[1], width / 2, 590); // 2 つ目のテキスト
      break;
    case 2:
      text(pairs[randomIndex].texts[0] + ( randomAmount / 100 ) + " 回 ", width / 2, 520); // 1 つ目のテキスト
      text(pairs[randomIndex].texts[1], width / 2, 590); // 2 つ目のテキスト
      break;
```

```
    // ボタンの残り回数を表示
    textSize(60); // テキストサイズを指定
    textAlign(CENTER, CENTER); // テキストの水平・垂直揃えをセンターに設定

      if (videoButtonRemainingCount > 0) { // 残り回数がある場合
    text(" 残り回数 ", width / 2, 1650);
    textSize(60);
    text(videoButtonRemainingCount + " 回 ", width / 2, 1730);
  } else { // 残り回数がない場合
    textSize(70); // テキストサイズを 34 に設定
    text(" 完了！ ", width / 2, 1730); // " 完了！ " を表示
  }

    textSize(40);
    fill(100,150,255,fadeAlpha); // テキスト色
    text(" 青色のボタン ", width / 2 - 255, 1500);
    fill(100,fadeAlpha); // テキスト色
    text(" を押して動物を痛めつけてください！ ", width / 2 + 125, 1500);



    fadeAlpha = min(fadeAlpha + 10, 255); // fadeAlpha を switch 文の外で更新

  if (videoButtonRemainingCount <= 0 && !delayStarted) {
    // 残り回数が 0 以下になったら、delayStarted が false の場合のみタイマーを開始
    videoStartTime = millis();
    delayStarted = true; // delayStarted を true に設定して、複数回タイマーが開始されないようにする
  }

  if (delayStarted && (millis() - videoStartTime >= videoDelayTime)) {
    // delayStarted が true かつ、遅延時間が経過したら
    enterPressCount = 5; // case 5 に移行
    caseStartTime = 0; // caseStartTime をリセット
    delayStarted = false; // delayStarted を false に戻す
  }

    break;



  case 5:
    // case 5 に入った時間を記録
    if (caseStartTime == 0) {
      caseStartTime = millis();
      effectPlayedInCase5 = false; // フラグを初期化
    }
    // エンター 5 回目：3 枚目の画像の上に新しい画像を表示
    image(img3, 0, 0, width, height);

    image(case5Images[randomIndex], 120, 720, 850, 830); // img3 の上に img5 を重ねて表示

    // 効果音を一度だけ再生
    if (!effectPlayedInCase5) {
     effects[1].jump(0); // 再生位置をリセット
     effects[1].play();   // 再生
      effectPlayedInCase5 = true; // 再生済みフラグを設定
    }


    textFont(font1);
    textSize(40);
    text(" かわいいたちの今日の痛めつけられ回数 ", width / 2, 520); // 合計金額を表示

    textFont(font2);
    textSize(60);
    text((totalAmount / 100) + " かい ", width / 2, 590); // 合計金額を表示

    textFont(font1);
    textSize(40);
    text(" 今日の合計金額 ", width / 2 - 120, 670); // 合計金額を表示

    textFont(font2);
    textSize(52);
    text( totalAmount + " 円 ", width / 2 + 120, 670); // 合計金額を表示



    textFont(font2);
    textSize(40);
    text(" もういじめないでね ", width / 2, 1650);

    // 動作しないと case0 に戻る
    if (millis() - caseStartTime >= case5DisplayTime) {
      enterPressCount = 0;
      caseStartTime = 0; // caseStartTime をリセット
      returnToStart = true; // リセット処理を実行
    }
    break;
  }
}

void keyPressed() {
  if (key == ENTER || key == RETURN) {
    if (enterPressCount == 1) { // enterPressCount == 1 の時のみ実行
      overlayVisible = true;
      fadeAlpha = 0;
    }
    enterPressCount++;  // enterPressCount を増やす処理を後ろに移動
    // 最後の画面に達したら、合計金額をリセットせずにループを開始
    if (enterPressCount > 5) {
      returnToStart = true; // リセット処理を実行
    }
  }
}
```

コード説明（processing）

このコードのポイントは、それぞれの画面ごとに
switch（enterPressCount) で case に
分けられている点である。
最初はエンターぼ回数で画面表示の制御をしていたのだが、
体験者に触ってもらいたかったので、ボタンや、カードリーダー
などの画面移行のトリガーとなるものを使うことにした。
あらかじめ switch（enterPressCount) で制御していた
おかげで、簡単に arduino のシリアル通信をコードに
組み込むことができた。しかし、case 分けをしていると、
動画にディレイをかけるにしても関数を作らないといけなくて、
大変すぎるのでもうこのやり方はやりたくない。
また、解像度の高い動画や画像を使いまくって全体で
5GB くらい使っているので、毎起動遅くてストレスだった。
あと、展示に際してモニターを研究室から借りたのだが、
まず縦画面表示が出来なくて詰み、プレゼンテーションモード
がうまくいかなくて詰み、そして解像度の調整をする羽目に
なってしまった。簡単に 90 度画面回転すると思ったから
借りたのに。

もう一つのポイントは、ボタンがいつ押されても動物の
動画が最初から再生されるところである。
前提として、動物の痛めつけられる映像は短い動画が
再生されていて、動画と動画の間は動画の始めと同じ画像
を表示している。
最初はボタンが押されると動画が再生し終わるまでボタンが
反応しなくて、テンポがすごく悪かった。
これをボタンを押した時に押した分だけ再生されるのという
コードを case4 の中だけで描いていたのだが、うまくいかず。
その後、ボタンを制御しているところでも同じコードを書いたら
なんとかうまく行った。苦節 1 週間。本当にむかついた。
最後のポイントは、動物たちに関するランダム関数である。
case ごとに分けているせいで、同じランダム関数を使っても
それぞれの case でランダム数字を生成しちゃうから、
前の case で作ったランダム数字を次の case に持ち越す
ことが大変だった。特に、動物関連のものは、ずれると
作品が成り立たないので絶対にずれがこないように
たくさん試行錯誤した。本当に頑張ったと思う。

やってみて思ったこと

今回、procesing と arduino を接続して、体験型の
決済アプリ『きゅーあぐ PAY』を制作したのだが、
コードの書いて、物も作って、デザインもして…と
やることが多すぎて最初は絶望しまくっていた。
何もかも一から自分で考えて生み出すことがまず難しいのに、
そこから展示物にまで仕上げるのって１ヶ月じゃ無理だろ！
と心の底から思っていた。
しかし、やってみると全てのことは繋がっているので、
これをやらないとこれが進まない…という自体に直面し、
結局なんやかんや終わる。ということがわかった。
私はコードを書くのも得意じゃないし、電子工作とか
今年からやり始めたのでハンダゴテは下手くそだけど、
出来ないことでもやりたいことをやってよかったー！
と完成した今になればそう思えている。
今回の制作で、自分のやりたいことが明確に定まったという
感触を得ることができたことが最大の収穫物だと思う。
展示できてよかったー！！

終わり