「ブロック崩し」の概要

「ブロック崩し」と本資料の位置付け

- 「ブロック崩し」
  - Monaca Educationのプロジェクトテンプレートの一つ
  - HTMLとCSS、JavaScriptで作られている
  - 動作の詳細はライブラリPIXI.jsに委ねている
  - プロジェクトにはソースコードが添付されている
  - ライブラリを利用することで、ソースコードは500行ほど
  - カスタマイズしてゲーム性を変えることができる
- 本資料
  - ソースコードの概要を説明する

ブロック崩し

プログラムの概要

JavaScript

- jsフォルダ
  - main.js
    - ゲームの本体
- componentsフォルダ
  - loader.js
    - PIXI.jsを読み込む
    - https://pixijs.com/

画像ファイル

- imgフォルダ
  - ボール、ブロック、パドルなどの画像ファイル

HTML

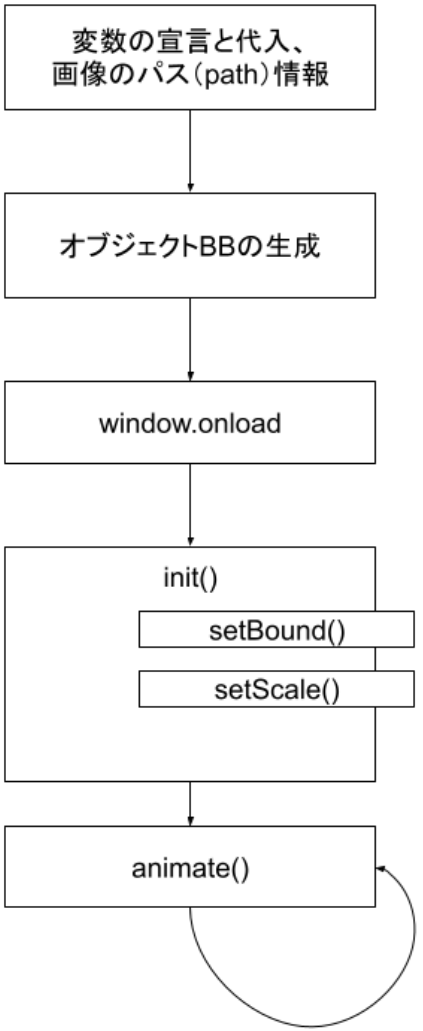
- index.html
  - ※このアプリでは、index.html は main.js や style.cssを読み込む以外の役割は無い

CSS

- style.css
  - 画面の基本色（背景色）や、canvas要素の表示方法を決めている

main.jsの流れ

1. 変数を宣言し、画像を準備する
2. ゲームの状態を表すオブジェクトBBを作る
3. 画面へのロードが済んだら、マウスやキーボードのイベントを処理できるように登録する
4. ゲームの状態を初期化する（画面のサイズを整える、得点をゼロにする、など）
5. 繰り返し、animate()が呼び出され、ゲームが進行する
    - 各瞬間のボールの状態、パドルの状態、ブロックの状態の判定をする



- main.jsは、名前の無い関数を宣言し、

  その場で呼び出す書き方

```
(
  function() {
    …
  }
) ();
```

- これをIIFE（即時実行関数式）と呼ぶ
- JavaScriptでは、関数の{ }の中に
  - 関数を書くことができる
  - オブジェクトを書くことができる
- main.jsも、関数の中に関数を書いている
- 修正するときには、カッコの対応に注意

main.jsのポイント

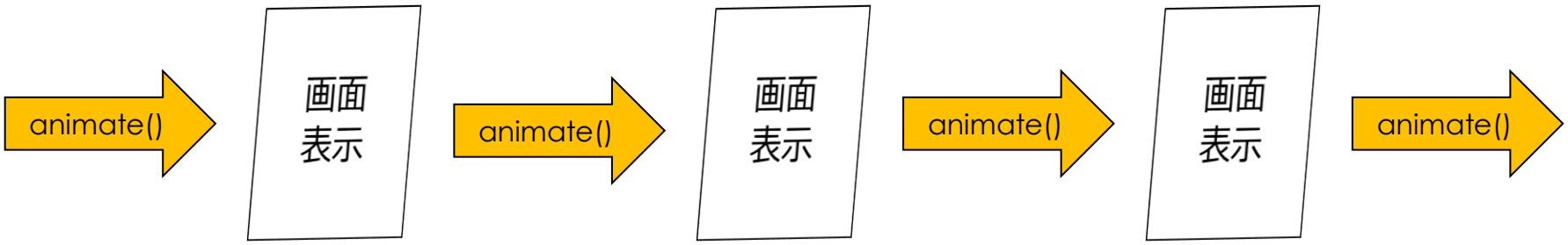
■ オブジェクトBB：ある瞬間のゲームの状態を保持するオブジェクト

- ゲーム内の要素（ボール、パドル、ブロック）の状態を保持するプロパティ
- それぞれの要素に対する操作を受けて状態の値を変更するメソッド
  - setMap：二次元配列による指定に従ってブロックを配置する
  - addBlock：setMapから呼び出される。１つ１つのブロックに、色・得点・画面上の位置（座標）を設定して、配置する
  - addBall：画面にボールを追加する。ボールの位置や速度の初期値を決めている
  - addPadle：画面にパドルを追加する。最初の位置や速度を決めている
  - addScore：得点を足す
  - reset：ゲーム開始時・ゲームリセット時に呼び出す。画面を初期化する

■ 関数animateは、BBのプロパティやメソッドを呼び出し、ゲームの状態を確認・更新して、次の瞬間のゲームの状態を作りだす

■ 画面の再描画前に実行される関数(1)

- この処理で、ある瞬間のゲームの状態が更新される
- 続いて、次の瞬間の画面が作られる
  - 最初は、関数init()の最後に、関数requestAnimFrameの引数として、requestAnimFrame(animate);の形式で指定している
  - 2回目以降は、関数animate()の最後に、自分自身を指定する



- if文で囲んであり、BB.gateStateがGAMESTATE_PLAYのときに実際の処理を実行する

■ 画面の再描画前に実行される関数(2)

- ユーザーの操作に合わせてパドルの位置を計算する
- 次の瞬間のボールの位置を計算する
  - 壁（左右の境界）に当たったら跳ね返す
  - 画面の下の境界より下に落ちたら、そのボールはゲームから消す
- ボールとパドルが当たっているか判定する
  - 当たっていたら、ボールの移動の方向を変える
- ボールと、残っているブロックが当たっているか判定する
  - 当たっていたら
    - 得点を加える
    - ボールの移動の方向を変える
    - ボールが当たったブロックを消す
    - 全てのブロックが消えていたら、ゲームクリアにする

資料

main.jsの内容

変数宣言と値の代入 （3行目～15行目）	アプリケーションの設定項目の値を決めている
画像のパス（path）情報 （19行目～27行目）	ボール、パドル、ブロックを表す画像ファイルに名前を付け、他の場所で利用しやすくしている
BBオブジェクトの生成 （30行目～281行目）	ある瞬間のゲームの状態を管理するオブジェクト。 animate()の中で、このオブジェクトの持つデータを利用する
初期化関数init() （284行目～341行目）	ゲームの準備処理を行う。 最後に、関数animate()を呼び出し、ゲームのプレイ処理へ進む
関数animate() （345行目～407行目）	次の瞬間に表示するゲーム画面（ボールの位置、パドルの位置、ブロックの当たり判定など）を計算する。 この関数は１秒間に数十回呼び出される。 最後に自分自身（関数animate）を呼び出し、ゲームのプレイ処理を繰り返す
アプリケーションがブラウザによって ロードされたときに呼び出される処理の指定 window.onload （409行目～415行目）	プログラムがブラウザによって読み込まれた直後、一番最初に実行される。 デバイスの準備ができたところで、 関数init()を呼び出す
関数setScale() （417行目～429行目）	ゲーム画面を作っているHTMLのcanvas要素の表示倍率を決める
関数setBound() （432行目～454行目）	ゲーム画面のサイズ（ボールが動く範囲。境界）を決める
関数vibrate() （459行目～462行目）	デバイスを振動させる
関数getUa() （464行目～472行目）	ユーザーエージェント（このアプリを動かしているデバイス）が何か調べ、iPhone/iPad/Androidなどデバイスを表す文字列を返す