素人がゲームを作れるのか？

著者　こうきゅうピーチ

初めに

「我々は生産する類のオタクではなく、消費する類のそれである。」

これは当サークルのある上級生の言葉である。我々のサークルは電子工作サークルを謳っておきながら、その実ゲームや漫画、アニメ鑑賞に興じるものになっている。もちろん実際に工作を行い、またプログラミングやパソコンに関する知識を持っている人々もいるにはいるらしいのだが、すくなくとも私自身はそうでもない。今回常盤祭に参加するにあたって、せっかくなら創作する側に回ってみよう、という軽い気持ちで始めたものを記したものが本著である。悪戦苦闘する初学者の模様をお楽しみいただけると幸いである。

今回簡単なゲームを製作するにあたって参考にしたのは「ゲームアルゴリズム　for JavaScript」（松浦健五郎/司ゆき著）である。部室の棚を適当に物色していたところ、最も表紙がポップだったので採用した。ちなみに今回作成したゲームのほとんどはこの教科書を模倣しており、オリジナル要素はごくわずかである。よもや本著を用いてプログラミングの練習を行う者など存在しないと思われるが、本著はあくまでゲーム製作日誌であり、コードや用語の説明は最小限度であることをお断りしておく。

HTML と JavaScript ってなに？

今回作成にあたり使用したプログラミング言語は JavaScript と HTML である。HTML（エイチティーエムエル、Hyper Text Markup Language）とはウェブページを作成するための言語であり、JavaScript はより詳細な記述を可能にする言語だ。ここではその違いはさして重要ではなく、HTML で作った箱に JavaScript をつかって中身を入れる、という理解で問題ない。両者を記述するのは簡単で、メモ帳などのアプリにコードを書き、それをそれぞれの拡張子というものをつけて保存する。拡張子はそのファイルがどのような形式であるかを決めるものである。、例えば「.txt」であれば単なるテキストであり、ただの文字列に過ぎない。これが「.png」であれば画像であり、「.mp3」であれば音声ファイルであることを示す。HTML は「.html」、JavaScript は「.js」とつけることで、それぞれの言語を記述したものとして機能する。

なお、著者はプログラミングに関してはほとんど素人であり、1 学年の秋学期に R 言語に触れたのみである。

基本的な書き方

上記のコードは、「human.png」と名付けられた画像をウェブ上に表示するものである。上が JavaScript で、下が HTML だ。HTML のコードでは、HTML のバージョンや JavaScript を使用すること、その規格やファイルが存在する場所などを記述している。JavaScript がコードの本体で、「image」というフォルダにある「human.png」という画像を表示することを示している。これくらい単純で簡単なコードでは HTML にまとめて記述するか、HTML のみを使って記述することもできるがのちのことを考えて分けて記述している。以降では、HTML は基本的に上記と変わらないため JavaScript のみ紹介する。

「コードは思った通りではなく、入力したとおりに実行される」とよく言われるが、そのとおりである。たった一文字やひとつの記号がないだけでもう受け付けてくれないので、たった一枚の画像を表示するためだけに筆者は一時間弱かかった。今回のゲーム製作に費やされた時間の大半はコードミスの捜索（俗に言う、バグつぶし）である。

この記事を見てゲーム製作にいそしもうとする人（？）のために、筆者が時間を費やしたコードミスを記しておく。

・セミコロン（；）の付け忘れ

セミコロンはコードの後ろにつけ、これからも次のコードが続くことを示す。これがないとそれ以降のコードを読み取ってくれない。最初の方のミスはたいていこれだった。コードが長くなると追加するたびに忘れそうになる。

・アポストロフィとクォーテーションマークのまちがい

音楽のファイルを指定するためにはアポストロフィ（’）を使うのだが、ほかのファイルを指定する場合にはクォーテーションマーク（“）を使う。ちょんちょんが一つか二つかの違いなのだが、筆者はホントに間違いに気づけずにふて寝した。

アニメーションをつくろう！

画像が表示出来たら、次はアニメーションである。今更説明する必要はないだろうが、アニメーションというのは複数の画像を連続で見せることで動きを出している。今回筆者がヒト型を動かすためにつかった画像はたった 4 枚だ。

↑今回自機として制作した人型。あまりにも雑である。ペイントツールで書いたが、動くようにするには結構大変だった。四枚目とか完全に足が折れているし、足の長さが毎シーン変わる奇形。でも意外と違和感ないから OK。

```
svar img;
var anim;
var animIndex;

function main() {
     img=document.createElement("img");
     document.body.appendChild(img);
     img.src="./image/human.png";
     anim=new Array(
          "human0.png","human2.png","human0.png",
"human1.png","human3.png","human1.png"
     );
     animIndex=0;
     setInterval(update, 50);
}

function update() {
     img.src=anim[animIndex];
     animIndex++;
     if (animIndex>=anim.length) {
          animIndex=0;
     }
}
```

以上がアニメーションを動かすためのコード。4 枚の画像を一定時間ごとに表示させ、最後まで来たらまた最初にループするように記述している。

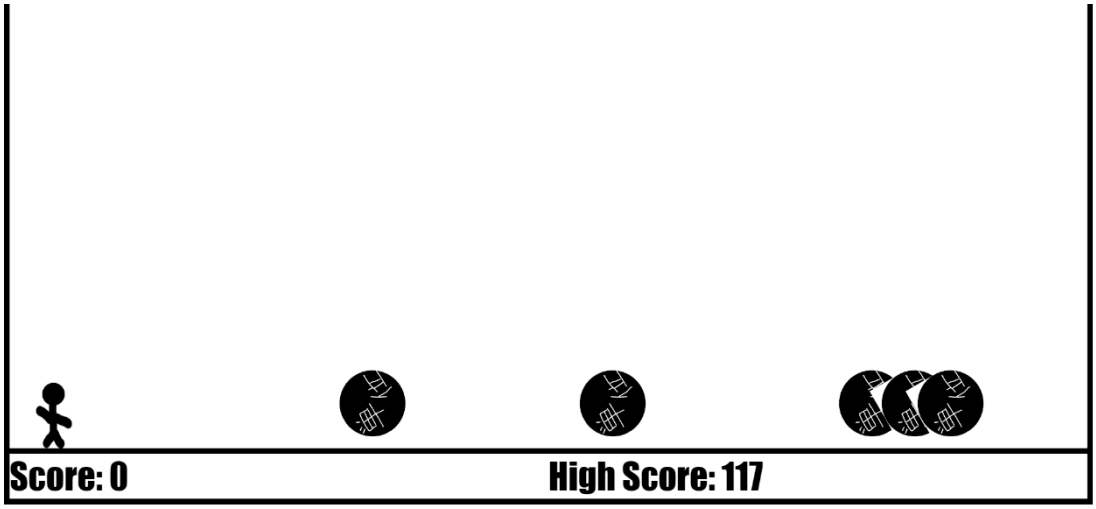
スクリプトをつくろう！

スクリプトというのは画像の集合体であり、実行されると背景とは独立したアニメーションを表示する。わかりにくい説明で恐縮だが、「オブジェクト」や、某マインクラフトでいう「エンティティ」が分かりやすいだろうか。

```
// スプライト
function Sprite(div, zIndex) {
	var obj=this;
	var style;

	// 表示領域
	obj.img=document.createElement("img");
	div.appendChild(obj.img);
	obj.img.src="void.png";
	style=obj.img.style;
	style.position="absolute";
	style.zIndex=zIndex;

	// 更新
	obj.updated=false;
	obj.viewX=0;
	obj.viewY=0;
	obj.onUpdate=function() {}
	obj.update=function(viewX, viewY) {
		if (obj.updated && viewX==obj.viewX && viewY==obj.viewY) return;
		obj.updated=true;
		obj.viewX=viewX;
		obj.viewY=viewY;

		var style=obj.img.style;
		style.left=Math.floor((obj.x-viewX)*CELL_SIZE)+"px";
		style.top=Math.floor((obj.y-viewY)*CELL_SIZE)+"px";
		style.width=Math.floor(obj.xSize*CELL_SIZE)+"px";
		style.height=Math.floor(obj.ySize*CELL_SIZE)+"px";

		obj.img.src=obj.animImage[obj.animIndex];
```

上記は今回作成したスプライトの設定の一部である。この下にこのスプライトのサイズや当たり判定、状態を付与する関数をつらつらと記述している。これだけでは共通の器ができただけであり、実際に動かすことや、その設定は個別に行う。

ゲーム本体を作ろう！



上記は、2 つの種類のスプライトを生成させたものである。一つは左端の人型で、これは自機である。キーボード操作によってジャンプと方向入力ができる。「単位」と書かれた五つの丸い物体が障害物で、左からランダムな速さと画面外の位置から右に流れてくる。人型が右にいるほど score と名付けた数字が加算される関数を定義・表示し、障害物にぶつかったら score はリセット。これが今回作成したゲームの原型である。

　なお上記の画像で使用した「単位」型障害物はボツ案である。最初は襲ってくる「単位」から逃げる、というゲームを作る予定だったが、文字を回転させるのがあまりにも難しかったので断念した（そもそも単位が転がってきたらみんな喜んで拾うよね～）。20 数枚もの画像を用意したのだが、なぜか画像の大きさがばらばらに表示されてしまうという事態が発生した。いまだにこの原因はわかっていない。だれか助けて。

ゲームのアレンジ

ここからは、よりゲームを快適に遊ぶためのカスタマイズをしていく。

・自機や障害物の変数の改良

```
var PLAYER_VX=0.2;
var PLAYER_VY=-1.0;
var PLAYER_AY=0.1;
var HURDLE_COUNT=4;
var HURDLE2_COUNT=1;
var HURDLE_SPEED_MIN=0.07;
var HURDLE_SPEED_MAX=0.5;
var HURDLE2_SPEED_MIN=0.03;
var HURDLE2_SPEED_MAX=0.06;
var HURDLE3_COUNT=1;
var HURDLE3_SPEED_MIN=0.1;
var HURDLE3_SPEED_MAX=0.4;
```

JavaScript の冒頭に、このような数字を定義している。上から順に自機の速さやジャンプ設定、障害物の数や速さだ。これらをいじることで自機が画面の上にぶっとんで当分帰ってこなかったり、爆速で障害物が右から左に流れたりもする。幾度のテストプレイを経て、最適なバランスを探ってみた。ゲームに慣れていない人は難しいかもしれない。

・ゲームオーバー時のアニメーション

移動とともに動く通常のアニメーションのほかに、ゲームオーバー時のアニメーションを追加する。画像を一定間隔で連続して再生することは同じだが、人型の状態を「生存」と障害物の当たり判定と衝突したときの「死亡」に分け、その「死亡」状態の時にアニメーションを再生するようにする。この時に、ゲームオーバー時の音声も再生する。

```
  obj.dead=function() {
         music2.play();
       obj.sprite.stepAnim();
       if (obj.sprite.isAnimOver()) {
            game.state=GS_continue;
       }
  }
```

・サウンドの追加

```
const music = new Audio('music.mp3');
window.addEventListener('keydown', function(){
music.currentTime=0;
music.play();
});
```

上記のように音楽ファイルを読み込ませ、それを決められたトリガーで再生する。このコードでは何らかのキーを入力したときに「music」と名付けられたファイルを再生するようにした。上から三行目の CurrentTime という関数で音

楽を再生されたときのタイマーをリセットし、連続でキーを入力しても絶え間なく音が鳴るようにしている。今回追加したサウンドは、上記とゲームオーバー時の２つである。

・中間画面の追加

ゲームの顔ともいうべきはタイトル画面である。今回追加した画面はタイトル画面、コンティニュー画面、レディー画面の 3 つだ。コンティニュー画面がない場合、一度ゲームオーバーになった後永遠に死亡し続けることになってしまう。

```
game.onUpdate=function() {
     switch (game.state) {
          case GS_title:
               inittitle();
               game.state=GS_title2;
          case GS_title2:
               title.stepAnim();
               if (game.isKey(KEY_ENTER)) {
                    initGame();
                    game.state=GS_ready;
               }
               break;
          case GS_ready:
               initSplash("ready.png");
               game.time=0;
               game.state=GS_ready2;
          case GS_ready2:
               game.time++;
               if (game.time==20) {
                    initSplash("void.png");
                    game.state=GS_game;
               }
               break;
          case GS_continue:
               initSplash("continue.png");
               game.state=GS_continue2;
          case GS_continue2:
               if (game.isKey(KEY_ENTER)) {
                    initSplash("void.png");
                    initGame()

                    game.state=GS_ready;
```

自機と同じように、ゲームにも状態を定義する。タイトル・レディー・ゲーム本体・コンティニューの 4 つである。ゲーム本体以外では、画面を遷移するための準備の状態をそれぞれ作成した。それぞれ決まったトリガーが実行されたとき（今回はエンターキーとゲームオーバー）にゲームの状態を変える。

・障害物のバリエーションの変化

今回は教科書の障害物のほかに 2 種類を追加した。左はパワーポイントによる自作で、右はフリー素材である。同じ動きをさせても面白くないので、左は鈍足/大きさ 2 倍、右は初期位置の y 軸をランダムになるよう設定した。左の障害物もアニメーションによって回転させたのだが、前述したように同じ大きさの画像群がなぜか大きさがバラバラに再生され、謎に奥行きが発生する問題が発生した。理由はわからないが、迫力があるので良しとした。

あとがき

筆者がプログラミングほぼ初学者にもかかわらずゲーム製作という企画に手を出してしまったので、果たして形になるのか心配だった。挫折した場合には一人旅紀行文かことしのプロ野球論評でも投稿しようかと腹をくくっていたが、これを読んでいる方がいるということはそれらは幻の記事になったのであろう。

今回作成したゲームは胸を張ってオリジナルといえる代物ではない（大半は教科書のコードを丸写ししただけ）が、できる限りオリジナリティやアレンジを加えようと努力はした。そんなゲームでも遊んでくれる人がいれば幸いである。改良の目標があるとするならばアイテムの概念の追加や、自機の改良（イラストや二段ジャンプ等の実装）だろうか。

最後に日ごろ好き勝手遊ばせてくれる愛すべき Scitex と歴代の偉大な先輩方に敬意と感謝を示して、この記事を終わりにしたい。