研究ノート

ゲームプログラミング向け教材コンテンツの開発

――JavaScript による「落ちゲー」を題材に――

河　村　一　樹

東京国際大学論叢　人間科学・複合領域研究　第 8 号　抜刷
2023 年（令和 5 年）3 月 20 日

研究ノート

ゲームプログラミング向け教材コンテンツの開発
――JavaScript による「落ちゲー」を題材に――

河　村　一　樹

Development of Educational Material Content for Game Programming
—The Subject of the “Falling Game” by JavaScript—

KAWAMURA, Kazuki

Abstract

The author has been in charge of programming education since 2013. The programming language used in the class was C, but switched to JavaScript in the middle of the class. The final task in the practical training was to create a game program from the construction of an original Web site. In line with this, the author decided to develop digital teaching materials for game programming. This paper focuses on the contents of the digital teaching materials. The content of the educational material is a step-by-step method, in which a JavaScript function is created for each step. By doing so, the goal is to eventually complete an original game program.

目　次

はじめに
1. プログラミング教育の変容
2. Web プログラミングからゲームプログラミングへ
　2.1　Web プログラミング
　2.2　ゲームプログラミング
3.「落ちゲー」の教材コンテンツの開発
　3.1「落ちゲー」のルール
　3.2　プログラムとしての仕様

3.3 「落ちゲー」の教材コンテンツ
おわりに

はじめに

プログラミング教育における実習では，さまざまな言語が使われている。当初はCを使っていたが，Cでは教授すべき内容が多く学生にとっては習熟レベルが高いので中途半端な学習到達度にしかならないこと，このため学生の学習に対するモチベーションが低下するといった問題が生じた。

そこで，Cのようなプログラミング言語ではなく，スクリプト言語であるJavaScriptに切り替えることにした。これにともない，Webの文章構造をHTMLで，レイアウトや装飾のデザインをCSSで，動きや操作といった動的表現をJavaScriptで，それぞれ実装するというWebプログラミングの実習を進めた。そして，最終的には，オリジナルのWebサイトの構築を目指すこととした。

その後，学期末の授業アンケートの結果から，動的なWebサイトが作りにくく，JavaScriptによるプログラミングがほとんどできないという意見が出てきた。そこで，実習の最終目標を，オリジナルなWebサイトの構築ではなく，ゲームプログラムの開発に変更することにした。この理由としては，JavaScriptがインタラクティブなゲームプログラミングに向いた言語であり，ゲームに興味関心を持つ学生が多いことから学習に対するモチベーションも向上することがあげられる。

ゲームプログラムにはさまざまなものがあるが，2次元のインタフェース上で動作する「落ちゲー」を取り上げ，そのプログラミングを学んでもらうための教材コンテンツを開発することにした。

教材コンテンツは，ステップアップ方式とし，ステップ毎にJavaScriptで新規に関数を作り込むか既存の関数を改変することとし，そのためのヒントについて取り上げた。これによって，学生自身がゲームプログラミングの実装方法について学び，実習の最終目標を遂行できるようになることを目指す。

1. プログラミング教育の変容

筆者は，ゼミナール（「「専門演習」以前は「演習（3）」「演習（4）」）や専門科目（「プログラミング基礎」以前は「プログラミング実習」，「ウェブアプリ論」）において，プログラミング教育を実践してきた。[1]

当初は，プログラミング言語としてCを採用していた。その理由として，厳密な構文規約にしたがってプログラミングができること，[2] 基礎から応用へと系統的に学べること，[3] 他大学でも採用実績が多いこと，[4] 教科書に良書[5]が多いこと，などによる。授業では，最初にCの構文規約やその使い方，および，基本的なアルゴリズムについて講義した。[6] 基本的なアルゴリズムとしては，文字列操作，数値計算，整列，探索，再帰などを取り上げた。その上で，各プログラムの例題に即した課題を用意し，それらについて，コンピュータを用いてプログラミングかつデバッグを繰り返すという実習を行った。

授業を進めるうちに，いくかの問題が顕著化してきた。一つは教育内容について，もう一つは学生の学ぶ姿勢についてであった。

前者については，基本的なプログラミングに終始してしまい，実用的なプログラムを開発する

スキルレベルまで到達できないという問題である。Cはシステム記述言語でもあり，オペレーティングシステムのような高度なプログラムを実装できるが，そのためには関数，ポインタ，アドレス，構造体，ガベージコレクションなどについての専門知識が必要になる。さらに，標準ライブラリ関数（文字列処理/文字処理，算術処理，一般ユーティリティなど）だけでなく，各種関数（時刻・日付管理，メモリ操作，疑似乱数操作，入出力など）の使い方についても理解していなければならない。つまり，敷居が高いプログラミング言語といえる。

後者については，学生にとって学習目標が定まらないという問題である。とくに，情報システム学科から情報ビジネス学科，そして，経営学科情報コースへと改組されてからは，学生の意識に変容がみられるようになった。それは，情報系の学習をしたいという学生が減ってきたということである。学科名に情報がついていた頃はプログラマ志望者も結構いたのだが，情報コースになるとプログラマを目指したいという学生がほとんどいなくなり，学生の志向が変わってしまった。その結果，学生は何のためにプログラミングを学習しているのかがわからなくなっている状況が見受けられるようになった。

一方，2000年代になってからは，インターネットの普及とともに，Webそのものが注目され始めた。その一環として，2004年から2011年までの間，World Wide Webに関する議論が集中的に行われたWeb 2.0 Summitが開催された。[7] ここで提唱されたWeb 2.0は，情報の受け手と送り手が固定されずに流動化し，誰でもがWebを介して情報を受発信できることを意味している。それを実現するサービスとして，検索エンジン，SNS，Wiki，BBS，blogなどがあげられている。

こういった状況に合わせて，Webの構築ができるスクリプト言語が登場してきた。スクリプト言語のスクリプト（script）は，「台本」という意味であり，簡易的に記述ができて可読性に優れた言語といえる。

また，プログラミング言語の多くはコンパイラ方式を採用しているが，スクリプト言語はインタプリタ方式になっている。コンパイラ方式ではソースコードを翻訳し連係編集を行った上でロードモジュールを生成して実行するが，インタプリタ方式ではソースコードを直接翻訳しながら実行する。つまり，インタプリタ方式では，プログラムを即実行できるという特長を持っている。

Webプログラミング向けのスクリプト言語としては，クライアントサイドとしてJavaScriptやPythonなどが，サーバサイドとしてPHP，Perl，Rubyなどがある。それらの中で，今回はJavaScriptを採択するに至った。

JavaScript（開発当初は，LiveScript）は，ネットスケープコミュニケーションズ社のブレンダン・アイク氏（Brendan Eich）により開発された。言語全体としては，制御構造を用いた手続き型だけでなく，クラスベースではなくプロトタイプベースによるオブジェクト指向型や高階関数の操作による関数型といったプログラミングスタイルを踏襲していることがあげられる。

JavaScriptの特徴としては，1）Cのように構文が複雑ではないのでプログラミングしやすい，2）HTML/CSSと連携した処理ができる，3）HTMLを直接操作・制御する，4）イベント駆動モデルを実装できる，5）コンパイラ言語のように専用のプログラム開発環境は必要なくエディタとブラウザだけあればよい，6）Webブラウザに言語処理系（インタプリタ方式，デバッガー）が組み込まれている，などがあげられる。

JavaScriptのプログラミングにおいては，フリーフォーマットであり記述に制約がないことや，言語仕様がゆるいこと[1]（変数を明示的に宣言しなくても利用できる，文字リテラルを囲む記号にシングルクォテーションあるいはダブルクォテーションのどちらでも使える，文の終端を表すセミコロン「;」がなくても構わない，同じ名前の関数を重複して宣言できるなど），ブラウザ（Internet

Explorer，Google Chrome）にデバッグ機能が組み込まれておりものがありF12キーでデバッガーを起動することができる。

以上の中で，JavaScriptを採択するに至った理由は，JavaScriptの特徴にある5）と 6）といえる。エディタでソースコードを入力してから搭載されているブラウザを起動すれば，即座に実行結果を確認できるからである。これによって，学生にとっても簡単に操作できるといったメリットが生じる。それだけでなく，通常エディタとブラウザはどのパソコンにも搭載されていることから，大学の実習室だけではなく自宅等でも実習ができる。これによって，学生に，自学自習ベースのプログラミングを奨励することもできるようになる。[8]

2. Webプログラミングからゲームプログラミングへ

2.1　Webプログラミング

CからJavaScriptに切り替えてからは，Webベースのプログラミングに取り組みことにした。Webプログラミングでは，クライアントサイドスクリプティングとしてHTMLとCSSとJavaScriptを取り上げることになる。Webページにおいて，HTMLは文章構造を作り，CSSはレイアウトや装飾といったデザインを行い，JavaScriptは動きや操作といった動的表現を行う。

（1）HTMLについて

HTMLでは，Webページにおける文書全体のデザインを設定することになる。表記の仕方は，

```
<!DOCTYPE　html>
<html lang=”ja”>
　　<head>
　　　<title>タイトル</title>
　　</head>
　　<body>
　　　<h1>タイトル</h1>
　　</body>
</html>
```

のようになる。

実習では，次のようなタグを取り上げた。[9]

・基本構造について

<html>：HTML文書，<head>：ヘッダー，<body>：本文，<!-- -->：注釈

・ヘッダー関連

<title>：タイトル，<meta>：文書情報，<link>：文書の前後関係，<script>：スクリプト言語

・ページレイアウト関連

<h1> 〜 <h6>：見出し文字，<hr>：水平線，
：改行，<p>：段落， ：センタリング， ：分割テキスト，<pre>：整形済テキスト，<blockquote>:ブロック引用，<address>:アドレス情報

・フォント関連

<font>：フォント，<i>：イタリック体，<tt>：等幅フォント，<b>：ボールド体，<u>：アンダーライン，<strike>：打消し線，<big>：大きいフォント，<small>：小さいフォント，<sub>：下付き文字，<sup>：上付き文字，：強調文字，：強い強調

・リスト関連

：番号なしリスト，<ol>：番号付きリスト，<li>：リスト項目，<dl>：定義型リスト，<dt>：定義語，<dd>：定義内容

・表関連

<table>：表組み，<tr>：表の行，<th>：表の見出し要素，<td>：表のデータ要素，<caption>：表題

・ハイパーリンク関連

<a>：アンカー

・イメージ関連

<img>：イメージ，<map>：イメージマップの指定，<area>：イメージマップデータ

・フォーム関連

<form>：フォーム，<input>：入力エリアの作成，<select>：選択メニュー，<option>：選択項目の指定，<textarea>：複数行テキスト入力フィールド

(2) CSSについて

CSSでは，「どこに」を表すセレクタ，「何を」を表すプロパティ，「どうするか」を表す値，をそれぞれ用いてデザインを定義することになる。表記の仕方は，

```
＜セレクタ＞{
    ＜プロパティ＞:＜値＞;
}
```

のようになる。

実習では，次のようなプロパティを取り上げた。[10]

・テキスト関連

font-size/weight/style/family：フォントのサイズ/太さ/スタイル/種類，text-align/decoration/indent/transform:水平方向の表示位置/文字の装飾/字下げ/大文字と小文字の変換，vertical-align：垂直方向の表示位置，letter-spacing：文字の間隔，word-spacing：単語の間隔，white-space：改行・スペース・タブの扱い

・色・背景関連

color：文字色，background：背景関連をまとめて指定，background-color/image/repeat/position/attachment：背景色/背景画像/背景画像の並び方/背景画像の表示位置/背景画面の固定表示

・幅と高さ関連

width：コンテンツの幅，height：コンテンツの高さ，max/min-width：最大/最小の幅，max/min-height：最大/最小の高さ

・マージンとパディング関連

margin-top/right/bottom/left：上下左右のマージン，padding-top/right/bottom/left：上下左右のパディング

・境界線関連

border-top/right/bottom/left：上下左右の境界線，border-top/right/bottom/left-width：上下左右の境界線の太さ，border-top/right/bottom/left-color：上下左右の境界線の色，border-top/right/bottom/left-style：上下左右の境界線のスタイル

・表示と配置関連

position：配置方法，top：上からの距離，right：右からの距離，bottom：下からの距離，left：左からの距離，z-index：重なりの順序，clear：回り込みの解除，overflow：はみ出した部分の

表示方法，display：表示形式，clip：切り抜き，float：フロート
・リスト関連
list-style-type/position/image：マーカーの種類/位置/画像
・テーブル関連
table-layout：表のレイアウト，table-collapse：境界線の表示方法，border-spacing：境界線の間隔，empty-cells：空セルの境界線，caption-side：表タイトルの位置
・アウトライン関連
outline-width/color/style：アウトラインの太さ/色/スタイル
・カーソル関連
cursor：カーソルの形

(3) JavaScript について

JavaScript では，Web ページにおいて，フォームに入力した値をチェックしたり，マウス操作に合わせて動的な振舞いを実装することができる。表記の方法は，html 内において，

```
<script>
    ステートメント;
</script>
```

とするか，外部ファイルに記述した file.js を html 内で呼び出し，

```
<script src="file.js"></script>
```

とするか，イベントハンドラで呼び出し，

```
<input type="button" onclick="alert('今日は')">
```

のようにする。

実習では，次のようなシンタックスを取り上げた。[11]
・数値/文字列/定数関連
10 進数，8 進数，16 進数，実数，浮動小数/文字の集まり/true，false，null，undefined，NaN
・配列関連
配列の宣言（new Array），配列の初期化（数値/文字），添え字（0 開始），配列の長さ（length），多次元配列，連想配列
・演算子関連
代入演算子（=），算術演算子（+，-，*，/，%，**，++，--），比較演算子（==，!=，===，<，<=，>，>=），論理演算子（&&，||），ビット演算子（&，|，^，~），複合代入演算子（+=，-=，*=，/=，%=），論理代入演算子（||-，&&=，??=），演算子の優先順位
・構文関連
複文（{…}），変数宣言（var/let），定数宣言（const），条件分岐（if else/switch），繰返し（while/do while/for），ループ中止（break），ループ継続（continue）
・関数関連
関数宣言（function），引数，戻り値，変数のスコープ（ローカル/グローバル変数）
・イベントハンドラ関連
オンクリック（onclick），オンキー（onkeydown/press/up），オンマウス（onmousedown/up/over/out/move），オンロード（onload/unload），オンサブミット（onsubmit/reset），オンインプット

（oninput），オンリサイズ（onresize），オンスクロール（onscroll）
・その他
セミコロン（;），コメント（//，/*から*/），予約語

以上のHTML/CSS/JavaScriptを用いて，プログラミングの実習を行った。実習課題については，上述したHTMLのタグとCSSのプロパティおよびJavaScriptのシンタックスを組み合わせた形で，一問毎に完結するようなプログラムとした。

授業の進め方については，それぞれの例題についてのレクチャーを行った上で，例題の応用的な問題としての課題を学生達が独力でプログラミングしながらデバッグを繰返す形をとった。そして，最終的にはオリジナルのWebサイト（例えば，仮想ショップ，ギャラリー，ドリル教材，クイズ集，…など）を開発することを課題として与えた。

学期末の授業アンケートにおいて，学生から次のような意見が出てきた。それは，オリジナルのWebサイトにインタラクティブな動きをつけにくいため，あまり動きのない通り一遍のものしか作れず，達成感が得にくいという指摘であった。この結果，HTMLとCSSによる開発が中心になってしまい，JavaScriptによるプログラミングがなかなかできず，学生のプログラミングに対する動機づけが低下してしまうという状況に陥った。

一方，プログラミングに関する学生の動機づけについては，学生が興味を引くプログラムとしては「インタラクティブなもの」が良い，[12] 段階的にプログラムを高度化できることがプログラム改良への動機づけとして有効である，[13] プログラミング言語を中心とした教育ではなく楽しくプログラミングが学習できる環境が必要，[14] プログラミングについて知るということを目指す場合もっと直接的で楽しめる動機づけを持った内容の教材が必要[15]という報告がある。そして，いずれの文献についても，ゲームプログラミングという共通キーワードがあげられる。また，ゲームプログラミングというと，JavaScriptによる実装事例が豊富にある。

以上のことより，プログラミング教育の到達目標を，オリジナルのWebサイト開発ではなく，ゲームプログラムの開発に変更することにした。ゲームプログラミングに関する課題を与えることで，学生は身近で「インタラクティブ」で「直接的」なゲームを題材に，プログラミングを「楽しく」学ぶことができるようになる。これによって，学生はゲームを完成させるという明確な到達目標を持つことができるとともに，プログラミングに関するモチベ―ジョンも維持できるといえる。

2.2　ゲームプログラミング

コンピュータゲームには，ゲームセンターにあるアーケードゲーム，専用のゲームマシンを使ったテレビゲーム，液晶ディスプレイを組み込んだ携帯型ゲーム，インターネットを経由したオンラインゲーム，パソコンを使うPCゲームなどがある。また，ゲームのジャンルには，アクションゲーム，シューティングゲーム，スポーツゲーム，リズムゲーム，パズルゲーム，ロールプレイングゲーム（RPG），落ち物ゲームなどがある。

HTML/CSS/JavaScriptによるプログラミングという視点からは，落ち物ゲーム（略称は「落ちゲー」，テトリス，ぷよぷよシリーズ，Dr.マリオなど）が適しているといえる。というのも，最新のHTML5からは，図形などのグラフィックスを描画できる要素としてCanvasが提供されている。Canvasは，JavaScriptから制御できるようになっており，グラフィックス関連の専用プロパティやメソッドも豊富に揃っていることから，JavaScriptによるゲームプログラミングに向いているといえる。

Canvas 要素を用いて描画をするためには，1）HTML で Canvas 要素を定義する，2）JavaScript で Canvas 要素への参照を取得する，3）Canvas 要素の参照からコンテキストを取得する，4）コンテキストに色や線の太さなどを設定する，5）コンテキストに対して線や四角形などの描画を行う，といった一連の操作を行う。[16]

例えば，Canvas を使って四角形を描く JavaScript によるプログラミングは，次のようになる。[17]

```
<!DOCTYPE html> <!--  HTML5  -->
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <script>  <!--  html 内に JavaScript のプログラムを記述  -->
      function canvas_test() {  <!--  オリジナル関数を宣言  -->
        var canvas = document.getElementById('test');  <!--  2) に相当  -->
        var ts = canvas.getContext('2d');  <!--  3) に相当  -->
        ts.strockeRect(20,20,80,80);  <!--  4) に相当  -->
      }
    </script>
  </head>
  <body onload="canvas_test()">  <!--  5) に相当  -->
    <canvas id="test" width="100" height="100">  <!--  1) に相当  -->
    </canvas>
  </body>
</head>
```

また，「落ちゲー」のプログラムでは，jQuery を使う場合と使わない場合の二通りのやり方がある。

jQuery とは，JavaScript のためのライブラリであり，これによってシンプルに JavaScript のコードを書くことができる。また，jQuery のライブラリには HTML や CSS の操作に関するコードが豊富に用意されており，それらを使うことで動的な振舞いを簡単なコードによって実装できる。

jQuery を読み込むには，Web サイトから読み込むか，公式サイトのダウンロードページから jQuery のソースコードをダウンロードする。jQuery の配布元に公開されたファイルを使用する（CDN）場合は，次のようなスクリプト文を記述する。

```
<script type="text/javascript" src="https://code.jquery.com/jquery-
3.5.1.min.js></script>
```

一方，公式サイトの URL は，https://jquery.com/download/ になる。

例えば，jQuery から HTML の DOM（Document Object Model）を取得する場合，基本的な構文は，

```
$("セレクタ");
```

と記述する。セレクタ部分には，CSS セレクタの同じ記法により，取得したい HTML 要素の検索条件を指定する。HTML 要素を jQuery で取得して操作する場合，HTML では，

```
<input type="text" id="Box" class="font-main">
```

とし，JavaScript では，DOM の id をキーにした場合は，

```
$("#Box");
```

DOM の class をキーにした場合は，

```
$(".font-main");
```

とすることで，jQueryからDOMを取得できる。[18]

以上のように，jQueryを使用するとなると，JavaScriptだけでなくjQueryも学ぶ必要が出てくる。そこで，今回はjQueryを使わずに，HTMLとJavaScriptだけを用いて「落ちゲー」をプログラミングすることにした。

3.「落ちゲー」の教材コンテンツの開発

3.1 「落ちゲー」のルール

「落ちゲー」では，画面の上部からランダムなブロックが落ちてきて，それを左右に動かしたり回転させることができる。ただし，左右の壁や積まれたブロックにめり込むような移動や回転はできない。ブロックは一定時間毎に下方向に落ちるが，ゲームが続くと落ちる速度が速くなっていく。ある程度早くなると，また元の速度に戻る。そして，横一列にブロックが揃えば，その行が消去され，得点がカウントアップされる。ブロックが，上部まで積み上がった時点で，ゲームオーバーとなり，そのときの得点を競うゲームである。

3.2　プログラムとしての仕様

教材コンテンツの作成にあたっては，JavaScriptによる「落ちゲー」のプログラミングに関して記述した文献[19]を参考にした（出版社には許諾済み）。

この文献の第6章から第7章で「落ち物パズル」を取り上げているが，各章の節毎にゲームプログラムの機能や動作を，JavaScriptを用いて実装しながらプログラムの完成に導くという流れになっている。

具体的には，1）効果音の設定，2）ゲーム画面の作成，3）ブロックの描画，4）ブロックの移動と回転，5）ランダムなブロックの描画，6）下移動と当たり判定，7）横一列に揃ったブロックの消去と得点，8）下移動の自動化，9）ゲームオーバーとなっている。このようなステップアップ方式によるプログラミングは，初心者にとってもわかりやすく学習効果が高いといえる。

3.1のルールをもとに，プログラムの仕様を詳細に記述すると，次のようになる。

(1) 初期画面

プログラムを起動後，最初に表示する画面である（**図1**）。

ゲームタイトル「落ち物パズル」は，画面左上からX方向に20px, Y方向に10px（以降，座標（20px, 10px）と表す）に位置づけ，<h1>とする。また，「得点：0」は幅380pxとする。

ブロックが落ちるゲームエリアは，座標（20px, 150px）に位置づけ，幅240，高さ440とする。

「（次ブロック表示エリア）」は，座標（300px, 150px）に位置づけ，幅80，高さ80とする。なお，図1には枠が表示されているが，ゲーム時には表示しない。

「ゲームスタート」は，座標（300px, 300px）に位置づけ，幅80，高さ50とする。

(2) 壁の描画

壁の中の背景は白色とする。壁を構成する一つの箱の大きさは20px × 20px，黒色で枠は白色の正方形とする。これを，左壁ではX=0の列に22個並べ，右壁ではx=11の列に22個並べ，下壁ではy=21の行に12個並べる（**図2**）。

(3) ブロックの種類

7つのブロックとする。ブロックを構成する箱1個の大きさは20px × 20pxの正方形とし。4箱×4箱で1ブロックを構成する。ブロックの形状（色彩コードは，#07C）は，**図3**のようにする。

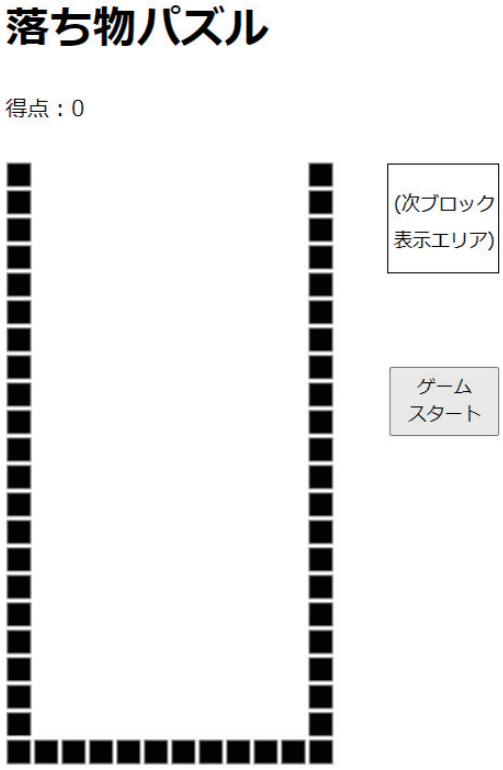

図 1　初期画面

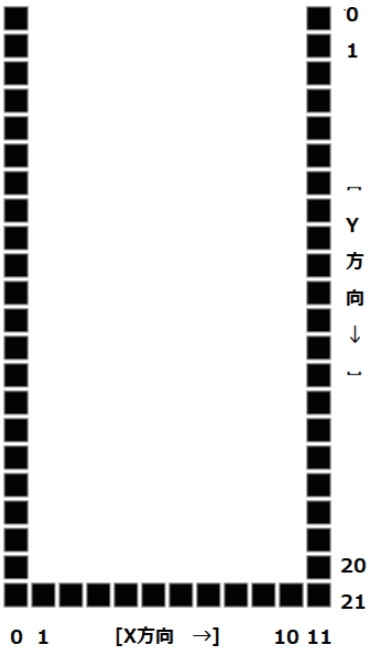

図 2　壁の構成

(4) ゲーム開始

「ゲームスタート」ボタンをクリックすることでゲームが始まる。ゲーム画面のX方向に4，Y方向に０の位置に，ランダムに抽出されたブロック（図3のどれか）を，「次ブロック表示エリア」にランダムに抽出された次のブロックを，それぞれ配置する（図4）

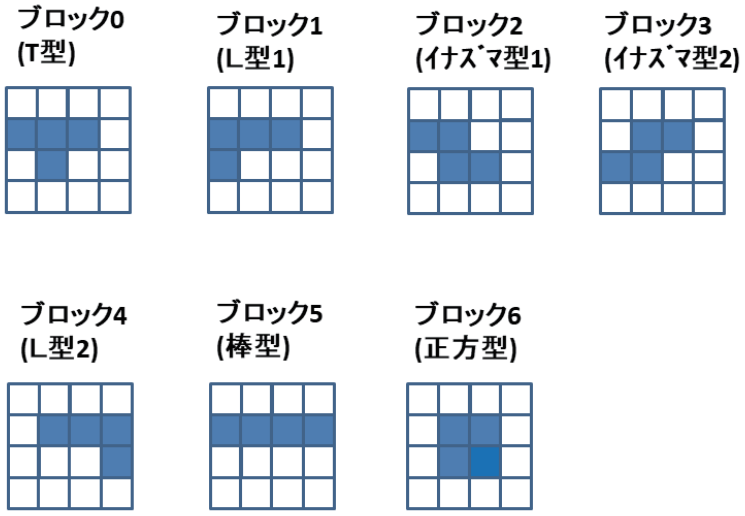

図３　ブロックの種類

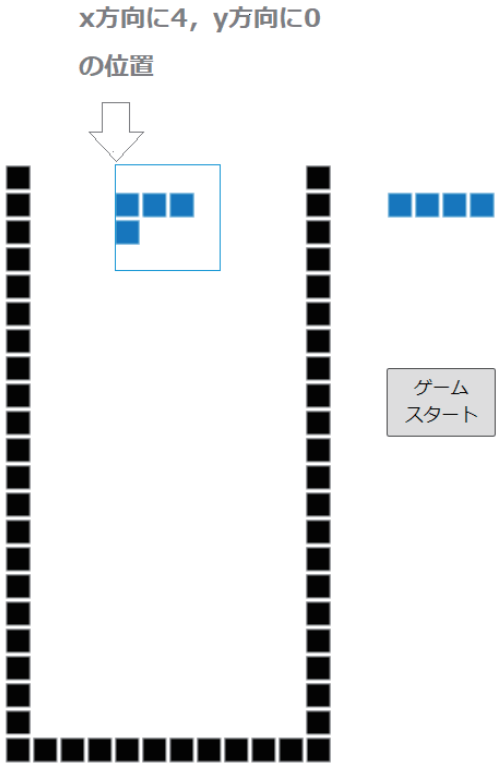

図４　ゲーム開始画面

(5) ブロックの左右移動

左矢印キー（←）か右矢印キー（→）を押すと，ブロック全体を1箱分だけ左か右に移動させる。ただし，壁や他のブロックにめり込むような移動はできない。

(6) ブロックの回転

上矢印キー（↑）を押すと，ブロック全体を回転させる。各ブロックの回転する順番（0回転は初回表示の状態）は，**図5**のようになる。ただし，壁や他のブロックにめり込むような回転はできない。

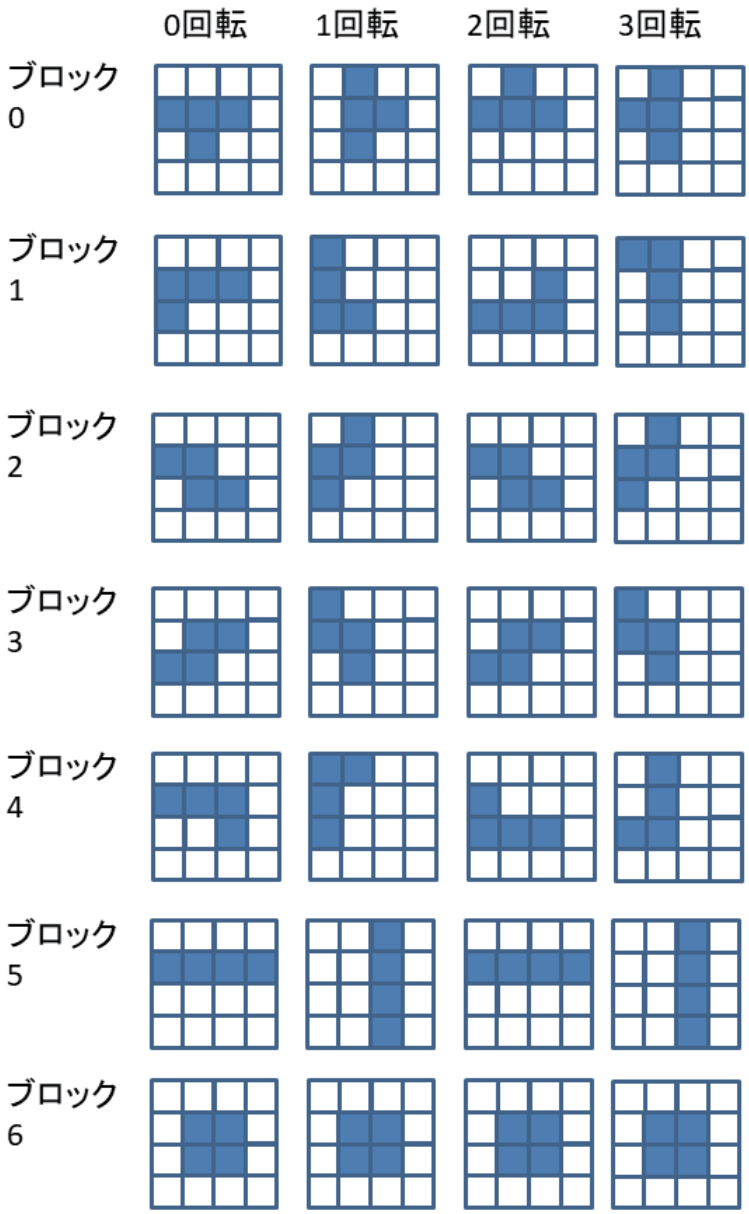

図５　ブロックの回転

(7) ブロックの下移動

下矢印キー(↓)を押すと，ブロック全体を1箱分だけ下に移動させる。ただし，壁や他のブロックにめり込むような移動はできない。

(8) 当たり判定

ブロックを移動したり回転するときに，壁やすでに置いたブロックと重ならないかを調べる。重なる場合は，移動も回転もせずに，元の状態に戻す。

(9) 横一列の消去と得点計算

各ブロックの箱が10個分横一列に揃ったときに，その行を消去するとともに，上部に置かれた全ブロックを下に移動させる。また，消去した行の数にもとづいて得点をカウントアップして表示する。具体的には，1行：10点，2行：20点，3行：100点，4行：(ボーナスポイントとして) 1000点とする。

(10) 自動で下移動

これには，タイマー機能を用いる。ゲームを開始する際には1秒から始めて，ブロックが落ちる速度は1ミリ秒ずつ早くなるように設定する。ある程度速くなる（50ミリ秒以下）と，また元の速度（1秒）に戻すこととする。

(11) ゲームオーバー

次のブロックを表示したときに，そこにもうブロックがあって重なってしまう場合はゲームオーバーとする。ゲームオーバーとしたら，アラート画面に「ゲームオーバー」と表示してプログラムを終了する。

(12) 効果音の付加

ゲームの操作中に，それぞれ効果音をつける。効果音としては，ブロックの左右移動と回転のときの音，ブロックの下移動のときの音，ブロックが底についたときの音，横一列が揃ったときの音，ゲームオーバーのときの音とする。

3.3 「落ちゲー」の教材コンテンツ

3.2のプログラム仕様に準拠した上で，教材コンテンツの開発を行う。開発の前提条件として，

・ステップアップ方式（オリジナル関数を随時追加）を採用
・htmlファイルは，こちら（筆者）で提供
・JavaScriptのプログラムは外部ファイル（program.js）とし，htmlから呼出す
・プログラムの各機能は，それぞれの関数として実装

とする。これより，ステップ毎に各オリジナル関数をJavaScriptでプログラミングすることになる。

教材コンテンツは，すべてMoodleにアップロードする。その際に，科目「プログラミング基礎」で実施している形と同じようにする。[20] 具体的には，

・[リソース]モジュールの[ファイル]：index.htmlファイル
・[活動]モジュールの[小テスト]（作文問題）：ステップ毎のプログラムの仕様
・[リソース]モジュールの[ファイル]：ステップ毎にJavaScriptでプログラミングする上でのポイント

とする。学生は，それらに基づきJavaScriptのプログラミングを行い，デバッグを終えてから，JavaScriptのソースコードを貼り付けるとともにprogram.jsを添付してアップロードする。

上述した「ステップ毎にJavaScriptでプログラミングする上でのポイント」についてまとめと，次のようになる。

【ステップ1：初期画面の表示】プログラム仕様（1）・（2）に準拠

●関数[＜初期画面＞()]の宣言[2)]

背景の Canvas を取得するには，

```
＜変数1＞=document.getElementById(‘back’); // HTML の id 要素 back を指定
＜変数2＞=＜変数1＞.getContext(‘2d’);
```

とする。壁の1つの箱を描画するには，

```
＜変数2＞.fillStyle=’#000000’; //箱の色（黒）
＜変数2＞.strokeStyle=’#ffffff’; // 箱の枠の色（白）
＜変数2＞.linewidth=3; // 箱の枠の幅
```

とする。左壁を描画するには，

```
x=0;
y=0;
for (i=0;i<22;i++) {
    ＜変数2＞.fillRect(x,y,20,20);
    ＜変数2＞.strokeRect(x,y,2020);
    y=y+20;
}
```

とする。右壁を描画するには，

```
x=220;
y=0;
for (i=0;i<22;i++) {
    ＜変数2＞.fillRect(x,y,20,20);
    ＜変数2＞.strokeRect(x,y,2020);
    y=y+20;
}
```

とする。下壁を描画するには，

```
x=20;
y=420;
for (i=1;i<11;i++) {
    ＜変数2＞.fillRect(x,y,20,20);
    ＜変数2＞.strokeRect(x,y,2020);
    x=x+20;
}
```

とする。

【ステップ2：ブロック0の描画（X方向4，Y方向0に固定）】プログラム仕様（4）に準拠

●関数[＜ゲーム開始＞()]の宣言

ゲーム画面（ブロックが動く領域）の Canvas を取得するには，

```
＜変数3＞=document.getElementById(‘game’); // HTML の id 要素 game を指定
＜変数4＞=＜変数3＞.getContext(‘2d’); /* ＜変数4＞に格納してオブジェクトがもつプロパティや
                                        メソッドは，＜変数4＞.＜プロパティ名＞，＜変数4＞.＜メ
                                        ソッド名＞で呼び出せる */
```

とする。前のゲームオーバーとなった画面全体を消去するためには，

```
＜変数4＞.clearRect(0,0,239,439);
```

とする。ここで，ブロック0の縦2横1の箱を描画するには，

```
＜変数5>=4; //X方向に4
＜変数6>=0; //Y方向に0
＜変数4>.fillRect(＜変数5>*20,(＜変数6>+1)*20,20,20); //X方向4＆Y方向1
＜変数4>.strokeRect(＜変数5>*20,(＜変数6>+1)*20,20,20);
```

とする。

同様に，ブロック0の縦2横2の箱を描画するには，

```
＜変数4>.fillRect((＜変数5>+1)*20,(＜変数6>+1)*20,20,20); //X方向5＆Y方向1
＜変数4>.strokeRect((＜変数5>+1)*20,(＜変数6>+1)*20,20,20);
```

とする。さらに，ブロック0の縦2横3の箱をX方向6＆Y方向1に，ブロック0の縦2横2の箱をX方向5＆Y方向2に，それぞれ描画する（**図6**）。

【ステップ3：ブロックの描画（関数化）】

ブロックを描画する処理を関数にすることで重複した記述が不要になり，必要な箇所に引数を用いて関数を呼び出すだけで済む。

●関数[<ブロック描画>(＜2dコンテキスト＞，<X方向>，<Y方向>)]の宣言[3)]

1つ目の箱を描くには，

```
<2dコンテキスト>.fillRect(<X方向>*20,(<Y方向>+1)*20,20,20);
<2dコンテキスト>.strokelRect(<X方向>*20,(<Y方向>+1)*20,20,20);
```

とする。2つ目の箱を描くには，

```
<2dコンテキスト>.fillRect((<X方向>+1)*20,(<Y方向>+1)*20,20,20);
<2dコンテキスト>.strokelRect((<X方向>+1)*20,(<Y方向>+1)*20,20,20);
```

とする。3つ目の箱を描くには，

```
<2dコンテキスト>.fillRect((<X方向>+2)*20,(<Y方向>+1)*20,20,20);
<2dコンテキスト>.strokelRect((<X方向>+2)*20,(<Y方向>+1)*20,20,20);
```

とする。4つ目の箱を描くには，

```
<2dコンテキスト>.fillRect((<X方向>+1)*20,(<Y方向>+2)*20,20,20);
<2dコンテキスト>.strokelRect((<X方向>+1)*20,(<Y方向>+2)*20,20,20);
```

とする。

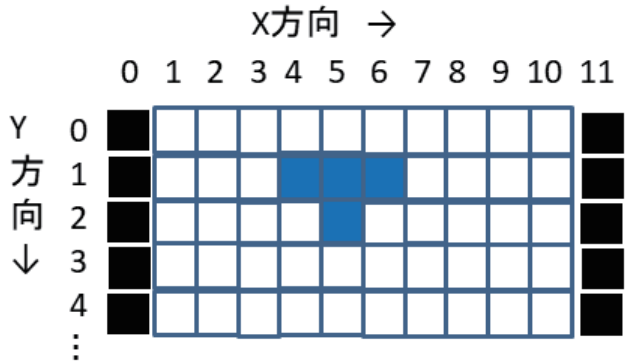

図6　ブロック0の描画

●関数[<ブロック消去>(＜2dコンテキスト＞，<X方向>，<Y方向>)]の宣言
　ブロックを消すには，

```
<2dコンテキスト>.globalCompositeOperation='destination-out'; /* 2dコンテキストが消去モードになり，描いた部分が消えるようになる */
関数[<ブロック描画>(＜2dコンテキスト＞，<X方向>，<Y方向>)]を呼ぶ /* 実際は消える */
<2dコンテキスト>.globalCompositeOperation='source-over'; // 元の描く状態に戻す
```

とする。
○関数[<ゲーム開始>]の修正
　ブロックの描画処理のところを，関数[<ブロック描画>]の呼び出しに変更する。

【ステップ4：ブロックの左右移動】プログラム仕様（5）に準拠
●関数[<ブロック移動>(<イベント>)]の宣言
　ステップ2と同様に，Canvasと2dコンテキストを取得する。次に，いまのブロックを消すために，関数[<ブロック消去>]を呼ぶ。そして，左・右矢印キー（← →）が押されたかどうかを判定するには，

```
if (<イベント変数>.keyCode==37) { // 左矢印キーが押されたとき
    <X方向>-=1; // 左に1つ移動
}
if (<イベント変数>.keyCode==39) { // 右矢印キーが押されたとき
    <X方向>+=1; // 右に1つ移動
}
```

とする。その上で，関数[<ブロック描画>]を呼ぶことで，新しい場所にブロックを描く。

【ステップ5：ブロックの回転】プログラム仕様（6）に準拠
○配列tblockの宣言
　ブロック0を描くパターンを，次のように配列（0は箱を描かず，1は箱を描く）として宣言する。

```
tblock=[
[ // ブロック0
[[0,0,0,0], // 0回転
 [1,1,1,0],
 [0,1,0,0],
 [0,0,0,0]],
[[0,1,0,0], // 1回転
 [0,1,1,0],
 [0,1,0,0],
 [0,0,0,0]],
[[0,1,0,0], // 2回転
 [1,1,1,0],
 [0,0,0,0],
 [0,0,0,0]],
[[0,1,0,0], // 3回転
 [1,1,0,0],
 [0,1,0,0],
 [0,0,0,0]]
];
```

○関数[<ブロック描画>]の変更

引数の4番目に，<向き>（ブロック0の場合，0回転はT型の下向き，1回転はT型の右向き，2回転はT型の上向き，3回転はT型の左向き）を追加する。また，配列から描画するパターンを決めるには，

```
<パターン>=tblock[<向き>]:
```

とする。次に，ステップ3ではブロック0を構成する4つの「1」について，それぞれfillRectとstrokeRectで描画したが，<パターン>と引数を使って描画するには，

```
for (n=0;n<4;n++) {
    for (m=0,m<4;m++) {
        if (<パターン>[n][m]==1) { // ブロック0の「1」の箇所を描画
            <2dコンテキスト>.fillRect((<X方向>+m)*20,(<Y方向>+n)*20,20,20);
            <2dコンテキスト>.strokeRect((<X方向>+m)*20,(<Y方向>+n)*20,20,20);
        }
    }
}
```

とする。

○関数[<ゲーム開始>]の変更

<向き>を0クリアーする。次に，関数[<ブロック描画>]の呼び出しにおいて，引数の4番目に<向き>を追加する。

○関数[<ブロック移動>]の変更

上矢印キー（↑）が押されたかどうかを判定するには，

```
if (<イベント>.keyCode==38) { // 上矢印キーが押されたとき
    <向き>=<向き>+1; // 回転する
    if (<向き>>3) {
        <向き>=0; // 3回転になったときは0回転に戻す
    }
}
```

とする。次に，いまのブロックを消すための関数[<ブロック消去>]の呼び出しにおいて，引数の4番目に<向き>を追加する。同様に，新しい場所にブロックを描くための関数[<ブロック描画>]の呼び出しにおいて，引数の4番目に<向き>を追加する。

○関数[<ブロック消去>]の変更

引数の4番目に<向き>を追加するとともに，ブロックを消すための関数[<ブロック描画>]の呼び出しにおいて引数の4番目に<向き>を追加する。

【ステップ6：ブロックのランダム描画】プログラム仕様（3）に準拠

○配列blockの宣言

ステップ5の配列tblockをblockに変えるとともに，ブロック0と同様にブロック1からブロック6まで追加する。

○関数[<ブロック描画>]の変更

引数の5番目に，ブロックの<種類>を追加する。指定されたブロックの種類で描くために，

```
<パターン>=block[<種類><向き>];
```

とする。

○関数[<ブロック消去>]の変更
　引数の5番目に，ブロックの<種類>を追加するとともに，ブロックを消すための関数[<ブロック描画>]の呼び出しにおいて5番目の引数に<種類>を追加する。
○関数[<ゲーム開始>]の変更
　ランダムにブロックを作るには，

```
<種類>=Math.floor(Math.random()*7);
```

とする。これによって，0から6の数値がランダムに生成され，その値をブロックの種類とする。また，関数[<ブロック描画>]の呼び出しにおいて，5番目の引数に<種類>を追加する。
○関数[<ブロック移動>]の変更
　関数[<ブロック消去>]と関数[<ブロック描画>]をそれぞれ呼び出すときに，5番目の引数に<種類>を追加する。

【ステップ7：次ブロックの描画】プログラム仕様（4）に準拠
●関数[<次ブロック設定>()]の宣言
　ステップ6と同様に，ランダムにブロックの種類を生成する。次のブロックを教示するには，

```
<変数7>=document.getElementById('tsugi'); // HTMLのid要素tsugiを指定
<変数8>=<変数7>.getContext('2d');
```

とする。次に，前のブロックを消去するには，

```
<変数8>.clearRect(0,0,79,79);
```

とする。そして，関数[<ブロック描画>]の5番目の引数に<変数7>を指定して呼び出す。
○関数[<ゲーム開始>]の変更
　関数[<次ブロック設定>]の呼出しを追加する。

【ステップ8：壁にめり込まない処理】プログラム仕様（8）に準拠
　ブロックを左右に移動した際に壁にめり込まないようにするためには，壁の位置をチェックし壁に重なるときは移動しないよう制御する必要がある。
○配列jyoutaiの宣言
　ブロックの重なり状態を把握するための配列を用意するには，

```
jyoutai=[];
```

とする。
○関数[<ゲーム開始>]の変更
配列jyoutaiをクリアするには，

```
jyoutai=new Array(22);
for (i=0;i<22;i++) {
    jyoutai[i]=new Array(12);
    for (j=0,j<12;j++) {
        jyoutai[i][j]=100; // jyoutaiを12列×22行の配列とし，全要素を「100」に
    }
}
```

とする。次に，左/右/下の各壁について，

```
for (i=;i<22;i++) {
    jyoutai[i][0]=99; // 左壁を「99」に
}
for (i-0;i<22;i++) {
    jyoutai[i][11]=99; // 右壁を「99」に
}
for (i=0;i<12;i++) {
    jyoutai[21][i]=99; // 下壁を「99」に
}
```

とする。

●関数[<確認>(<X方向>,<Y方向>,<向き>,<種類>)]を宣言

ブロックのパターンを決めるには，

```
<パターン>=block[<種類>][<向き>];
```

とする。次に，4×4のブロックの<パターン>の中で「1」の箇所が，配列jyoutaiにおいてX方向およびY方向が壁の中で「100」であれば動かすことができ（true），そうでなければ動かせない（false）を設定するには，

```
for (n=0;n<4;n++) {
    for (m=0;m<4;m++) {
        if (<パターン>[n][m]==1) { // ブロックの中の「1」の箇所にいて
            if ((<X方向>+m<0)||(<X方向>+m>11)) {
                return false; // X方向が範囲外のときは移動できず
            }
            if ((<Y方向>+n<0)||(<Y方向>+n>21)) {
                return false; // Y方向が範囲外のときは移動できず
            }
            if (jyoutai[Y方向+n][X方向+m]!=100) {
                return false; // 空欄（「100」）でないときは移動できず
            }
        }
    }
}
return true; // 上記以外の場合は移動できる
```

とする。

○関数[<ブロック移動>]の変更

現在のブロックの座標（<X方向>，<Y方向>）と向き（<向き>）を保存してから，移動・回転ができるかどうかを確認するために関数[<確認>]を呼び出す。その戻り値（trueかfalse）をkekkaに代入し，

```
if (!kekka) { // 移動や回転ができないとき
    保存したブロックの座標と向きを戻す
}
```

とする。そして，現在の座標と向きと種類に基づいて関数[<ブロック描画>]を呼び出し，ブロックを描画する。

【ステップ9：ブロックの下移動】プログラム仕様（7）に準拠

●関数[＜ブロック下移動＞()]の宣言

描画するためのCanvasを取得するには，

```
＜変数9＞=document.getElementById(‘game’); // HTMLのid要素gameを指定
＜変数10＞=＜変数9＞.getContext(‘2d’);
```

とする。次に，現在の座標（X方向とY方向）と向きを保存し，これらを引数として関数[＜ブロック消去＞]を呼び出す。そして，Y方向の座標だけに1を加え，関数[＜ブロック描画＞]を呼び出す。下に移動できるかどうかを確認するために，関数[＜確認＞]を呼び出し，戻り値をkekkaに代入し，

```
if (kekka) {
    そのまま関数[＜ブロック描画＞]を呼び出す // 下に移動できる
} else {
    移動前の座標と向きに戻して関数[＜ブロック描画＞]を呼び出す
}
```

とする。それとともに，この位置をあたり判定をするための配列jyoutaiに設定するために，

```
＜パターン＞=block[＜種類＞][＜向き＞];
for (n=0;n<4;n++) {
    for (m=0;m<4;m++) {
        if (＜パターン＞[n][m]==1) {
            jyoutai{＜Y方向＞+n}[＜X方向＞+m]=＜種類＞; /* 次のステップの横1行消去に使用する
                                                        ため */
        }
    }
}
```

とする。次のブロックとして設定したものが下に落ちるようにするには，<X方向＞を4，<Y方向＞を0，＜向き＞を0，＜種類＞をランダムに生成した次のブロックの種類として関数[＜ブロック描画＞]を呼び出す。そして，関数[＜次ブロック設定＞]を呼び出す。

○関数[＜ブロック移動＞]の変更

下矢印キーが押されたときには，

```
if (＜イベント変数＞.keyCode==40) { // 下矢印きー　が押されたとき
    関数[＜ブロック下移動＞]を呼び出す
}
```

とする。

【ステップ10：横1行消去と得点計算】プログラム仕様（9）に準拠

○関数[＜ゲーム開始＞]の変更

得点（tokuten）を0クリアーする。

●関数[＜得点計算＞()]の宣言

すべての行について，横方向に揃っているかを調べるには，

```
kosuu=0; // 消去した行数を0クリアー
for (y=0;y<21;y++) {
    sorottenai=false: // 揃っている状態で初期化
    for (x=1;x<=10;x++) {
```

```
            if (jyoutai[y][x]==100 { // 99（壁）でもなく，ブロック（0から6）でもない
                sorottenai=true; // 揃っていない状態
            }
        }
```

とする。行が揃っているときは，

```
    if (!sorottenai) {
        kosuu+=1; // 消去した行数をカウントアップ
    for (k=y;k>0;k--) { // 上から順に1行ずつ詰めてコピーすることでその行を消去
        for (x=1;x<=10;x++) {
            jyoutai[k][x]=jyoutai[k-1][x];
        }
    }
```

とする。次に，'game'のCanvasを取得するには，

```
    <変数11>=document.getElementById('gamei'); // HTMLのid要素gameを指定
    <変数12>=<変数11>.getContext('2d');
```

とする。ゲーム画面全体を消去するには，

```
    <変数12>.clearRect(0,0,239,4399);
```

とする。その上で，ブロックのあるところを描くには，

```
    for (y=1;y<21;y++) {
        for (x=1;x<11;x++) {
            if (jyoutai[y][x]!=100) {
                <変数12>.fillStyle='#07c';
                <変数12>.strokeStyle='#ffffff';
                <変数12>.lineWidth=3;
                <変数12>.fillRect(x*20,y*20,20,20);
                <変数12>.strokeRect(x*20,y*20,20,20);
            }
        }
    }
```

とする。得点計算では，kosuuの値が1ならば得点（tokuten）に10点を，2ならば20点を，3ならば100点を，4ならば1000点をそれぞれ加算する。その得点を表示するには，

```
    <変数13>=document.getElementBYId('tokuten'); // HTMLのid要素tokutenを指定
    <変数13>.innerHTML=tokuten;
```

とする。

○関数[<ブロック下移動>]の変更

関数[<得点計算>]を呼び出す

【ステップ11：ブロックの下移動の自動化】プログラム仕様（10）に準拠

ブロックを自動的に下に動くようにするためには，タイマー機能を使う。

○関数[<ゲーム開始>]を変更

ゲームが実行中のときに実行し，タイマーをセットするには，

```
jikkou=true; // ゲームを実行中と設定する
jikan=1000; // タイマーの時間をミリ秒単位で指定する。初期値は1秒とする
setTimeout（関数[<時間移動>],jikan）; // 1秒毎に，関数[<j時間移動>]を実行する
```

とする。

●関数[<時間移動>()]の宣言

実行中であれば，

```
if (jikkou) { // 実行中
    関数[<ブロック下移動>]を呼び出す
    setTimeout(関数[<時間移動>],jikan); // 再帰呼び出し
}
```

とする。

○関数[<ブロック下移動>]の変更

下に移動する時間を徐々に速くするには，

```
jikan-=1; // 1ミリ秒ずつ速くする
if(jikan<50) {
    jikan=1000; // 50ミリ秒未満になったら，1秒に戻す
}
```

とする。

【ステップ12：ゲームオーバー】プログラム仕様（11）に準拠

○関数[<ブロック下移動>]の変更

次のブロックを表示したときに，他のブロックと重なるかどうかを調べるには，関数[<確認>]を呼び出し，戻り値をkekkaに代入する。その値をもとに，ゲームオーバーかどうかを判定するには，

```
if (kekka) {
    alert(‘ゲームオーバー’); // アラート画面に「ゲームオーバー」と表示
    jikkou=false; // タイマーの処理によって下に動かすのをやめる
}
```

とする。

【ステップ13：効果音の付加】プログラム仕様（12）に準拠

○効果音については，HTMLファイルにおいて，

```
<audio id="kaiten" preload="auto"> <source src="oto/kaiten.mp3" type="audio/mp3">
</audio>  <!-- ブロックの移動や回転のときの音 -->
<audio id="ochiru" preload="auto"> <source src="oto/ochiru.mp3" type="audio/mp3">
</audio>  <!-- ブロックが一つ下に動くときの音 -->
<audio id="don" preload="auto"> <source src="oto/don.mp3" type="audio/mp3">
</audio>  <!-- ブロックが底についたときの音 -->
<audio id="kieru" preload="auto"> <source src="oto/kieru.mp3" type="audio/mp3">
</audio>  <!-- ブロックが横1行揃って消去するときの音 -->
<audio id="zenbu" preload="auto"> <source src="oto/zenbu.mp3" type="audio/mp3">
```

```
</audio> <!-- ブロックが4行分揃ったときのボーナス音 -->
<audio id="gameover" preload="auto"> <source src="oto/gameover.mp3"
 type="audio/mp3"> </audio> <!-- ゲームオーバーのときの音 -->
```

とする。

○関数[<ブロック下移動>]の変更

それぞれの処理で効果音をつけるには，

```
// ブロックを移動するところで
document.getElementById('ochiru').play();
// ブロックが底について移動できないところで
document.getElementById('don').play(); //
// ゲームオーバーとなったところで
document.getElementById('gameover').play(); //
```

とする。

○関数[<ブロック移動>]の変更

それぞれの処理で効果音をつけるには，

```
// 右/左/上矢印キーがぞれぞれ押されたところで
document.getElementById('kaiten').play();
```

とする。

○関数[<得点計算>]の変更

それぞれの処理で効果音をつけるには，

```
// ブロックが揃っているところで
document.getElementById('kieru').play();
// 得点計算で4つ分揃っているところで
document.getElementById('zenbu').play();
```

とする。

おわりに

以上，ゲームプログラミング「落ちゲー」の教材コンテンツの内容について記載した。

今後の予定としては，3.3をもとに開発する教材コンテンツをMoodleに実装するとともに，その教材コンテンツの学習効果を測定するための実証実験を行う。学生がステップ毎にprogram.js（仕様通りの完成版が前提）をMoodleにアップロードしたときの日時をログデータとして取得することで，学生さらにはクラス全体の課題達成度合いを把握することができる。

最終課題として，学生のスキルレベルに応じて，オリジナルのゲームプログラムの作成か，既存のゲームプログラムの改変を与えることにする。これによって，総合的なゲームプログラミングのスキルを判定することができるようになる。

また，学生がゲームプログラミングにどの程度関心があるのかについても事前調査を行う。これによって，学生の関心度合と課題達成度合いの比較を行うことにする。

注

1）「“use strict”;」を記述すると，より厳密なエラーチェックが可能になる．

2）これは，「function＜関数名＞() {…}」のことで，引数なしの関数を宣言している．
3）これは，「function＜関数名＞(＜引数＞,…){…}」のことで，引数ありの関数を宣言している．

参考文献

[1] 河村一樹：情報ビジネス学科におけるプログラミング教育，東京国際大学論叢商学部編，第81号，pp. 23–39，2011年.
[2] ISO/IEC 9899:2011/Cor 1:2012, International Organization for Standardization
[3] 江見圭司：C言語の歩き方――Windows環境における系統的なC言語教育，コンピュータ＆エデュケーション，Vol. 8，pp. 51–56，2000年.
[4] 岡部成玄：一般情報教育の全国実態調査（2），情報処理，Vol. 56，No. 1，pp. 96–100，2015年.
[5] Brian W.Kernighan, Dennis M.Ritchie : The C programming language. Second Edition, 1988.
[6] 河村一樹，非品正照：文科系のためのプログラミング論，日刊工業新聞社，2000年.
[7] History of Information.com : Web 2.0 EXPO, 2004
https://www.historyofinformation.com/detail.php?id=1654
[8] 河村一樹：Moodleを用いた自学自習ベースのプログラミング教育，東京国際大学論叢人間科学・複合領域研究，第6号，2021年.
[9] ふれあいインターネット：HTMLタグ一覧
http://www.fureai.or.jp/~irie/html-tag/
[10] TAG index：CSSプロパティ一覧
https://www.tagindex.com/stylesheet/properties/
[11] 杜甫々：とほほのJavaScriptリファレンス
https://www.tohoho-web.com/js/index.htm
[12] 小林健一郎：プログラミング教育におけるゲームプログラム，静岡産業大学情報学部研究紀要，15，pp. 1–16，2013年.
[13] 中村　孝：ゲームプログラミングを題材としたプログラミング教育の試み，情報処理学会「情報教育シンポジウム」2001論文集，9号，pp. 23–26，2001年.
[14] 栗山　裕，橋下友茂，山下利之：ゲームプログラミングによる情報教育の評価方法，日本教育工学会論文誌 28（Supple.），pp. 181–184，2004年.
[15] 森田信一，大滝真知子：一般情報処理教育でのプログラミングに関する一考察――ゲーム制作からLOGOのプログラミングを学ぶためのモデルカリキュラム――，十文字女子短期大学研究紀要，第27集，pp. 45–48，1996年.
[16] 田中賢一郎：ゲームで学ぶJavaScript入門，インプレス社，2015年.
[17] 中島省吾：HTML5版HTML演習，SCC，2017年.
[18] TechAcademyマガジン：今さら聞けない！ jQueryとは
https://techacademy.jp/magazine/9444
[19] 大澤文孝＆できるシリーズ編集部：できるきっず子どもと学ぶJavaScriptプログラミング入門，インプレス社，2018年.