3章 インタラクティブコンテンツエンジン

3-2 enchant.js と enchantMOON が目指すもの

清水 亮†

キーワード：Education, Visual Language, JavaScript, Game Development, User Interface, End-user Programming

1. プログラミングとは何か

プログラミングとは何か。またはプログラムとは何か、という問いをわれわれは忘れてしまいがちである。プログラムの歴史がコンピュータの発明より遥か以前にあるということを、現代の人々はほとんど意識していない。英語の program の語源はギリシャ語の *προγραμμα* に由来し、これは「公に書かれたもの」という意味であった。プログラムとは、単にコンピュータを操るためのものではなく、極言すれば、「自分の居ないところで、自分以外の知的存在を自分の思い通りに動かすための手順書」である。ファーストフードの接客マニュアルも広義にはプログラムと呼べるし、母親が子供を躾けるのも、一種のプログラミングと呼べる。

電子計算機が登場して半世紀、時代は1台のコンピュータを複数の人間が操るメインフレーム時代から、ダウンサイジングによる一人1台のパーソナルコンピュータ化が完了し、一人の人間が無数のコンピュータを同時に操る世界、すなわち、ユビキタスコンピューティング時代が到来しようとしている。計算機の処理能力は年々向上し、意識的なプログラミングなしでもかなり高度なことができるようになってきているが、仮にプログラミングがビデオデッキの録画予約程度にまで簡易化されれば、人類は新しい次元へ一歩進むことができるであろう。すなわち、地球上のすべての人物がプログラミングできる機械に囲まれた世界の到来である。筆者らの究極の目標は、その世界の実現にある。

2. プログラミング言語とは何か

われわれ人類は、今日、無数と言えるバリエーションのプログラミング言語を有している。そのバリエーションの増加は止まるところを知らない。メジャーと呼べる言語ですら数百はあるであろう。なぜわれわれはこうも多くのプログラミング言語を必要とするのであろうか。日本語や英

語などの自然言語も、本来はプログラミング言語として用いることができる。なぜならば、自然言語からC言語、Rubyなどのプログラミング言語とは可逆的に変換できるからである。換言すれば、人工プログラミング言語とは、人間がプログラミングをしようとするときに見落としがちな論理構造上の矛盾をあぶり出し、矛盾なくプログラミングを行うための道具であると言える。

本来、生まれ持った自然言語に近い形でプログラミングできるほうが便利であるが、自然言語には非論理的であったり不明瞭であったりする部分があるため、どうしても、厳密な意味でのプログラミングに直接利用することはできない。また、プログラミング言語として用いられるのは主に英数字を中心としているが、より自然言語に近い状態で使用できるはずの欧米においても、プログラミング言語を扱うことができるのはごく一部の訓練された人間だけになっている。すなわち、プログラミング言語は、自然言語に似せていても、明らかに自然言語ではなく、それ故に一般の人々から敬遠される傾向がある。

プログラミング言語を習得するのがどの程度困難なのか、筆者らはいくつか実証実験を行った。

まず東京大学の文科系学生を集めて行ったプログラミング講習では、配列の概念を説明するところで行き詰まった。配列をうまく扱うにはループの概念も知る必要があるが、配列とループを仮に教えたとしても、それを実際に何に使うのか、ということがまったく伝えられないのである。結局、抽象的な説明を繰り返すほかなく、これは脱落者が多かった。

次に、電通において、社内で選ばれた若手エリート20名にプログラミングを教えてみた。彼らの場合、最初につまづいたのは[]や[]といった、プログラミング言語で使う記号の入力方法であった。また、この括弧の対応関係を飲み込むのにも相当な時間を要し、結局、ループと配列の概念をその場で覚えられた者は一人もいなかった。一方で、オブジェクト指向は非常に多くの人々がすぐに飲み込めた。これは人間のもともとの機能として、他者になりかわって想像する機能が備わっているためと思われる。また、おそらく同様の理由で、イベント駆動についてもすぐに理解さ

* 本稿の著作権は著者に帰属致します。

† 株式会社ユビキタスエンタテインメント
"The Ultimate Goal of enchant.js and enchantMOON" by Ryo Shimizu
(Ubiquitous Entertainment Inc., Tokyo)

れた。座標の概念は基礎的な数学で出て来るので、これも問題はなかった。

プログラミングをより多くの人に教示するために、筆者らはゲーム開発を題材とし、まず実際のゲーム開発フレームワークをJavaScriptで実装することにした。これがenchanted.jsである。

3. enchanted.js

～高速開発を可能にするフレームワーク～

enchanted.jsは、HTML5ベースのゲーム開発フレームワークで、特に無駄なく高速に開発できることを目指した。独自のクラス構造を持っており、プロトタイプベースのオブジェクト指向機能しか持たないJavaScript上で、クラスベースのプログラミングを実現でき、イベント駆動モデルでプログラムを記述する。

典型的なenchanted.jsの入門プログラムは以下ようになる。

```
enchanted();
window.onload=function(){
  game = new Game(320, 320);
  game.onload = function(){
    label = new Label("Hello World");
    label.x = 100;
    label.y = 100;
    label.addEventListener("ontouchend",
    function(){//イベントリスナーを定義
      label.x+=100;
    });
    game.rootScene.addChild(label)
  }
  game.start();
}
```

イベント等の用語はAdobe Flashに似せた。enchanted.jsのリリース時点ではAdobe FlashのActionScriptのプログラマがenchanted.jsのプログラマの想定ユーザだったため、用語を似せることで移行をしやすくした。見て解るように、enchanted.jsのプログラム自体は極めてシンプルである。やりたいことをひたすら書くだけでプログラムが完成するようになっている。

ちなみにクラスを使ったプログラムは以下ようになる。

```
Bear = Class.create(Sprite, // Sprite クラスを継承
{
  initialize: function(x, y){//初期化する
    Sprite.call(this, 32, 32); //Spriteオブジェクトを
    初期化
    this.image = game.assets['chara1.gif'];
    this.x = x;
    this.y = y;
    this.frame = 0;
    game.rootScene.addChild(this);
  },
```

```
//enterframe イベントのリスナーを定義する
onenterframe: function(){
  this.x++; //右へ移動
},
//touchend イベントのリスナーを定義する
ontouchend: function(){
  this.frame += 3; //泣く
}
});
```

```
window.onload = function(){
  game = new Game(320, 320);
  game.preload('chara1.gif');
  game.onload = function(){
    bear1 = new Bear(10, 100); //一人目のクマさん
    bear2 = new Bear(40, 110); //二人目のクマさん
    bear3 = new Bear(0,120); //三人目のクマさん
  }
  game.start();
}
```

enchanted.jsが目指したのは、仕様書を書くようにプログラムを書く、それが完成している、ということである。したがって、JavaScriptの特徴を最大限活かしてできるだけ短い行数でゲームプログラミングが完了するよう工夫されている。3Dプログラミングや物理シミュレーションのような高度なプログラミングも、enchanted.jsは非常に短い行数で記述できるようになっている。enchanted.jsの容易さは非常に多くの人々に受け入れられ、リリースからわずか10日で100本近い作品が作られ、登場から2年経過した今日までに3,000本以上の作品が開発されている。商用サービスにも多数採用された。

4. puppet.enchanted.js ～仕様書を書くようにプログラミングする

しかし、enchanted.jsは一定の成果を収めたものの、すべてのあまねく人々にプログラミングの叡智を教示するという目標には遠く及ばない。そこで、より仕様に近い形でプログラムを記述できるよう、puppet.enchanted.jsというプラグインを開発した。

puppet.enchanted.jsによるプログラムは次のようになる。

```
Puppet.create("くま",{
  filename: "chara1.png",
  w: 32,
  h: 32,
  behavior:["standAlone": "zigzagX"]
});
```

これだけで、画面の中央にクマの絵が表示され、左右にゆらゆらと動くプログラムになる。"standAlone"と"zigzagX"はともにpuppet.enchanted.js側にもっている「ビヘイビア（振る舞い）」という要素で、ユーザが増やすことも

できる。

例えば zigzagX の中身は以下のようにになっている。

```
zigzagX:{
  enterframe: function(){
    this.x += Math.cos(this.age * 0.1)* 0.1 *
    this.speed;
  }
},
```

puppet.enchant.js のビヘイビアは、イベントリスナーの塊である。処理をイベントリスナーの塊にすることによって、さまざまな特徴を自由に付け加えることができる。zigzagX の仲間 zigzagY というものがある、これの中身は以下のようにになっている。

```
zigzagY:{
  enterframe: function(){
    this.y += Math.sin(this.age * 0.1)* 0.1 *
    this.speed;
  }
},
```

この二つを組合せた以下のようなプログラムを書くこともできる。

```
Puppet.create("くま",{
  filename: "chara1.png",
  w: 32,
  h: 32,
  behavior:["standAlone": "zigzagX", "zigzagY"]
});
```

この、ビヘイビアの合成というパラダイムは、puppet.enchant.js の特徴の一つでもある。自然言語で仕様書を書く場合、例えばそのキャラクタはどんな絵か、ファイル名とキャラクタのサイズ、そしてそのキャラクタはどのように振る舞うか、それだけが記述されていることが多い。puppet.enchant.js はまさしくその仕様を記述すると、その通りに振る舞うようにプログラムされる。また、puppet.enchant.js で解決しようとしたのは、ループと配列の問題である。題材をゲーム開発に限定すれば、ループと配列はほとんどの場合は排除できる。なぜならば、ゲーム開発において、ループと配列が出て来るのは、初期配置の部分と複数のキャラクタを扱ったり、当たり判定を扱ったりする場合に限られるからである。puppet.enchant.js では当たり判定もビヘイビアで定義する。

その定義は以下になる。

```
Puppet.create("くま",{
  filename: "chara1.png",
  w: 32,
  h: 32,
  behavior:["randomSetup",
    "zigzagX",
    "hitAndDie",
  ]
},
```

```
init: function(){
  this.collision.push("くま");
},
hit: function(){
  score +=10;
}
});
```

behavior 配列に文字列ではなくオブジェクトを与える、それはビヘイビアをインライン定義することになる。当たり判定は hitAndDie というビヘイビアと、その直後の init イベントリスナーで this.collision という当たり判定用コレクションに、当たり判定を行うクラス(この場合、"くま")を指定するだけである。その後に定義されている hit イベントリスナーが当たった場合に呼ばれるようになる。内部的にはコレクションをループして指定されたクラスのインスタンスと当たり判定を行い、当たった場合に hit イベントを発生させている。プログラミング習得の初期段階でループや配列を見せないことによって、仕様書を書くようなプログラミングを実現している。ただし、もっと簡易的にするのなら、collision は init イベントで呼ぶのではなく、プロパティとして定義するべきだろう。このような記述になっているのは、後述するビジュアル言語 MOONBlock と関係がある。

5. MOONBlock ～キーボードスキル不要のビジュアル言語から本格的なプログラムへの橋渡し～

puppet.enchant.js によってプログラミングはさらに簡略化、高速化されることになった。プログラマは個別の動きをアルゴリズムに落とし込まなくても、簡単に成果だけを使うことができる一方、自分自身の手でビヘイビアのバリエーションを追加することができ、なおかつ他のビヘイビアとの合成までできる。しかし明らかな欠点は、これはキーボードを使って記述する言語であるということである。電通社員への研修の結果、明らかになったのは、プログラミング言語を操るのは通常自然言語の文章を入力するのとは異なるキーボード操作スキルが要求されるということであった。また、括弧の対応など、人間が頭で処理しなければならない複雑な要素がからみ、これが学習のための障害となっていた。そこでキーボードスキルが未発達な人間でもプログラミングができるように、ビジュアル言語を用意した。それが MOONBlock である。これは当初、開発に影響を与えた人物の名をとって前田ブロックと呼んでいたが、MOONBlock という呼称に統一された。MOONBlock と puppet.enchant.js は同時に開発したため、collision の扱いが奇妙になってしまった。

先ほどの当たり判定のプログラムを MOONBlock で記述すると、図1のようになる。これはかなり仕様書に近い記述ができるようになっている。

図1 MOONBlockによるシンプルなプログラム

ビジュアル言語の先行事例として、ScratchやSqueak Etoysがある。この二つの言語は、どちらもプログラミング教育を目的としている点でMOONBlockと似ているが、MOONBlockの目的はあくまでも、実用的なプログラミングを広くあまねく人々に解放することであり、最終目標は実用言語として長く利用されることであるのに対し、Scratch, Squeak Etoysはともにプログラミングの入門までをサポートし、その後はより専門的な言語にステップアップすることを前提としている。この学習ギャップは大きく、Scratchで学習をした子供が、ステップアップしてenchant.jsを学ぼうとするとやはりキーボードスキルでつまずく他に、それまでとはまったく異なる（しかしプログラミングの世界では一般的な）パラダイムに頭を切替える必要が生じて混乱する。

MOONBlockでは、ビヘイビアとして動きを宣言的に追加する他に、イベントリスナーを手続き言語的に追加することができる。また、この部分にJavaScriptを直接書くこともできる。学習者は、このプロセスを経ながら、任意のタイミングで自らの配置したブロックが、どのようにJavaScriptに変換されているか確認することができる。

```
Puppet.create("くま",{
  filename: "chara1.png",
  w: 32,
  h: 32,
  behavior:["randomSetup",
  {
    sceneStart: function(){
      this.interval = 30;
      this.initialNumber = 10;
    }
  }
}:"hitAndDie",
{
  init: function(){
    this.collision.push("くま");
  }
},{
  hit: function(){
    enchant.puppet.Theatre.instance.score +=
      parseFloat(0) | 0;
  }
}
```

図2 イベントの記述

図3 実行ブロックでは直接JavaScriptを記述可能

```
},{
  enterframe: function(event){
    this.x += ~(Math.random()* 20)- 10
  }
}
});
```

こうして、JavaScriptのプログラミングへ誘うようステップアップが用意されているのも特徴である。

6. enchantMOON～未就学児から老人までプログラミングできる端末を目指して

こうしたプロセスを経て最終的に筆者らが現在開発・販売しているのがenchantMOONと呼ばれるコンピュータ製品である。これはハンドルのついたタブレット形状をしているが、エンドユーザがプログラミングすることを前提とした独自OS"MOONPhase"を搭載している。ペンディジタルイザとキャパシティブタッチセンサを両方内蔵し、ペンで手書きし、指で描いた図形を操作するという機能を持っている。特筆すべきは、まずキーボードをまったく操ることができなくても、絵を描き、描いた絵の一部分を他のページにハイパーリンクすることができるようになっている点である。これなら未就学児童でも自然にハイパーテキストを作ることができる。また、ハイパーリンクにした絵（これをシールと呼ぶ）を指で囲むと、「Hack」ボタンが現れ、Hackボタンを押すとMOONBlockのビジュアルプログラ

ミング環境が立ち上がる。MOONBlockが立ち上がると、そのシールのプログラムの中身が表示される。この中身は、ハイパーリンクを貼った直後の場合、ハイパーリンク先にジャンプする、という内容のプログラムが書かれている。これをカスタマイズすることで、例えばペンの色や太さを設定するとか、ゲームをプログラムしたりすることができるようにになっている。他のユーザが作ったシールも、Webブラウザからダウンロードしてインストールすることができる。これらの技術はすべてenchant.jsが基板となっており、突き詰めるとenchant.jsによるプログラミングまでのステップがすべて用意されていることになる。

また同時に、enchantMOONでは描かれた内容をシールプログラム側から操作したりできるようになっている。これによって、Undo機能や罫線を引く機能、日付を挿入する機能などが実現されている。

また、複数のアプリケーションを一つのアプリケーションとして合成する機能の搭載も検討している。イベントリスナを記述するだけという完全なイベント駆動パラダイムのため、複数のシールを合成するには単に順序を与えて呼び出せば良いということになるためである。アプリケーションの吐き出すデータを合成するパイプ処理ではなく、アプリケーションそのものを合成するという試みは、珍しいのではないかと考えている。例えばユーザは、好みの罫線と好みの配色のシールを合成し、自分だけのシールプログラムを作ることができるというわけである(図4)。

7. その先に目指すもの

プログラミングはそれほど難しくない。当初、筆者らが想像していたよりも、プログラミングは簡単にできることが解った。事実、小学生や未就学児がenchantMOONで作品作りをしたり、それまでプログラミングと聞くだけで難しそうと思って手を出さなかった人々が、enchantMOONをきっかけにプログラミングする楽しさや喜びに目覚めたりしている。いずれプログラミングができるということが特別な能力を意味しなくなるだろう。

筆者が想定しているのは、料理である。料理は、かつて非常に重労働であった。しかし不可欠なものであったため、訓練を積んだ者が炊事を行った。一つ一つの手順、包丁の扱いから火加減、さじ加減まで、すべてを覚えるには非常に手間が掛かった。しかし時代が進むと、瓶詰め、缶詰、電子レンジ、レトルト食品、冷凍食品といったものが増え、料理は特別な手間をかけなくてもできる時代が到来した。コンビニエンスストアにいけばいつでも弁当とおにぎりが手に入る時代である。その結果、いまや料理は生活能力として不可欠なものではなく、趣味の一つとなった。もちろん手間ひまをかけて素晴らしい料理を作る料理人は職業として依然存在するが、料理は今や生活能力として求められる必須的な能力ではなくなった。

プログラミングも、時代が進めば、いずれそうなるだろうと筆者は考えている。ビジュアル言語やアプリケーショ

図4 リンクを作っただけのシールの中に予めプログラミングされた内容

ンの合成といったことは、いわば冷凍食品やレトルト食品である。それでも、コンビニ弁当しか選択肢がないよりはずっといい。

既製品のアプリケーションというのは、いわばコンビニ弁当のようなものである。一見すると奇麗で、楽しげで、魅力的だが、実際には食べたいものだけが入っているわけではなく、望まない防腐剤も入っているかもしれない。けれども利便性をこの価格で実現するにはコンビニ弁当には保存料が必要で、コストはシビアにならざるを得ないのである。既存のアプリケーションだけに頼って生きる時代は、本来不健全なのである。アプリケーションはバージョンアップ料金を徴収して会社が生き延びるという、ただそれだけの目的のために機能を肥大化させ、リソースを圧迫し、やがて新しいコンピュータが必要になる。

そのような時代は、20世紀で終わりにしようではないか。人々が、自分の頭で、必要な機能を判断し、必要な機能を選びとり、自ら使う道具を自らの手で育てて行く。そういう時代を目指して、筆者らは研究開発を続けている。

(2013年11月11日受付)

清水 亮 (しみず りょう) 1998年、電気通信大学在学中に米Microsoftにて、同社のコンシューマ機向けミドルウェア事業「WindowsCE for Dreamcast」の日本国内展開を担当する。翌年、同大学を中途退学。1999年、(株)ダウンゴで携帯電話事業の立ち上げを担当。2002年、米シアトルのDWANGO North America社コンテンツ担当副社長に就任。2003年、(有)ユビキタスエンタテインメントを立ち上げる。2005年、(独)IPAより天才プログラマ/スーパークリエイターとして認定される。