2005－EC－2（11）
2005／12／17

部屋型コンテンツの制作を支援するシステムの開発

吉田 直史　櫻沢 繁　松原 仁
公立はこだて未来大学

概 要

コンピュータ上にて仮想の部屋を表現する部屋状のエンタテインメントコンテンツは，ユーザがオリジナルの部屋をレイアウトできるという点で多くの評価を得ている．このような部屋型のコンテンツはユーザによる情報表現の一つとして有用であり，エンタテインメントに限らず幅広い応用が期待される．しかしながら，従来のシステムではコンテンツの用途が限定されており，ユーザが独自に部屋の利用法を考案することまではできなかった．そこで本研究では，用途を限定しない部屋型コンテンツを制作しインタラクティブに動作させるためのシステムを開発し，様々な用途を持ったオリジナルの部屋型コンテンツを簡単に制作するための環境を構築した．

Development of a support system for creating room-style contents

Naofumi Yoshida, Shigeru Sakurazawa and Hitoshi Matsubara
Future University-HAKODATE

Abstract

The system for creating a virtual room expressed on a computer is supported by many users cause the flexibility of a room layout. These systems are expected to be useful tools for user's self-expression. However, the usage of existing systems which create room-style contents tends to be limited and the user cannot devise an original usage of user's own room. So, we developed the flexible system creating original rooms and running interactively. In this paper, we describe the system overview and introduce some examples.

1　はじめに

近年，コンピュータ上に仮想の部屋を構築し，その中に人や動物を模したキャラクタを生活させるようなエンタテインメントコンテンツが数多く提供されており，子供から大人まで多くのユーザが利用し，部屋のレイアウトやキャラクタの育成を楽しんでいる．このようなカスタマイズ可能な部屋型のコンテンツは創造性が高く人を惹きつける魅力があり，ユーザによる情報表現の一つとして今後幅広い応用が期待される．しかしながら，従来の部屋型コンテンツのシステムはキャラクタを育てることが目的であったり，部屋を飾り立てることが目的であったりと，誰もがわかりやすく簡単に利用できるように部屋の利用目的が開発者によって限定されており，部屋の拡張性には多くの制約があった．

そこで本研究では，用途を限定しない部屋型のコンテンツを制作し，キャラクタをインタラクティブに動作させるためのシステム，Interactive Doll House System (IDHS) を開発し，様々な用途を持ったオリジナルの部屋型コンテンツを簡単に制作するための環境を構築した．本論では，この IDHS の概要，及び IDHS によって制作可能なコンテンツの例を紹介し，このシステムの特徴について述べる．

2　部屋型のコンテンツ

本論では，コンピュータ上にて表現されるインタラクティブな部屋状の空間を部屋型コンテンツと呼ぶこととする．一般に流通している部屋型コンテンツのシステムとして代表的なものに Sim People [1]，Habbo Hotel [2] などがある．

Sim People は家を作り，そこに人を住まわせ育成することを目的としたコンピュータゲームであり，実世界の家のように土地を買い家の外観を設計し，室内の内装をレイアウトすることができる．また Habbo Hotel は，サーバから提供される仮想の空間に自分の部屋を作り，内装をレイアウトして他のユーザを呼びキャラ

クタ同士でチャットを行うためのオンラインコミュニティサービスである．Sim People のシリーズは全世界で 2000 万本以上を売り上げ，また Habbo Hotel は北欧で 500 万人のユーザを記録しており，両者共に多くの支持を得ている．両者の用途は異なるものの自分好みのオリジナルの部屋を制作できるという点で共通しており，部屋のカスタマイズ性がユーザを惹き付ける要因の一つであると考えられる．

しかし，これらの部屋型コンテンツは既存の家具や壁紙，キャラクタの外観などのリソースを用いて部屋をレイアウトすることはできるが，ユーザが独自に部屋のリソースを開発することができないためにユーザによる拡張性が低く，そのために従来の部屋型コンテンツシステムは開発者側のサポートに左右される一過性のサービスとして終わることが多かった．

そこで本研究では，部屋のリソースの多くをユーザがカスタマイズ可能で，さらにキャラクタの振る舞いを自由にプログラミングすることが可能な拡張性の高いインタフェースを構築し，ユーザ自身が利用法を考案できるような部屋型コンテンツシステムを提案する．

3 IDHS

IDHS は XML ドキュメントと 3D オブジェクトデータとで構成された部屋のデータとキャラクタのデータとを読み込み，キャラクタ間のインタラクションを処理し 3D 空間にレンダリングするシステムである．キャラクタのインタラクションは C# のクラスライブラリとして記述し，IDHS のデータ読み込み時に.NET Framework のリフレクション機能を用いて呼び出す．IDHS は C#，.NET Framework，Managed DirectX とによって実装されている．

IDHS では部屋型コンテンツを構成する要素を，床，壁，家具，キャラクタとし，家具やキャラクタをモデルと呼び，これらを組み合わせた部屋全体をシーンと呼ぶこととする．IDHS でのインタラクションはモデル間の相互作用，またはモデルとシーンとの相互作用のことを示す．シーンはグリッド状に分割されており，そのグリッドの 1 格子をセルと呼ぶ．モデルの移動や他モデルとの接触判定の処理はセル単位で行う．

3.1 シーンデータ

シーンデータを定義する XML ドキュメントの例を図 1 に示す．まず scene タグにてシーンの名前やサイズを指定し，次に床の配置，壁の配置を floor，wall タグにて指定する．またシーンデータでは天井の有無や，背景，カメラの座標も指定できる．最後にシーンに配置するモデルを models タグ内にて指定してシーンデータが完成する．配置可能なモデルの数は自由であるが，床が存在するセルにのみ配置することが可能で，1 セルにモデルを複数配置することはできない．モデルのデータは別の XML ドキュメントとして外部に記述する．

```
<scene name="kumaroom" width="4" depth="4">

  <floor texturepath="floor.jpg">
    1,1,1,1,
    1,1,1,1,
    1,1,1,1,
    1,1,1,1
  </floor>

  <wall texturepath="wall.jpg" height="1.4">
    1:1, 1:0, 1:0, 1:0, 0:1,
    0:1, 0:0, 0:0, 0:0, 0:1,
    0:1, 0:0, 0:0, 0:0, 0:1,
    0:1, 0:0, 0:0, 0:0, 0:1,
    1:0, 1:0, 1:0, 1:0, 0:0
  </wall>

  <ceiling texturepath="sky.jpg" fillmode="fit" />

  <background texturepath="bg.jpg" r="130" g="180" b="250" />

  <camera>
    <position x="0.0" y="4.0" z="-8.0" />
    <target x="2.5" y="0.0" z="-2.5" />
    <fov degree="45" />
  </camera>

  <models>
    <model path="kuma.model.xml" position="0,0" direction="3" />
    <model path="chabudai.model.xml" position="2,2" />
  </models>

</scene>
```

図 1: シーンデータの例 (一部)

3.2 モデルデータ

モデルデータを定義する XML ドキュメントの例を図 2 に示す．まず model タグにてモデルの名前を指定し，それからインタラクションを記述したアセンブリのパスとタイプを指定する．アセンブリ指定が無ければモデルは動かない静的な 3D オブジェクトとして描画される．次に graphics タグ内にてモデルの 3D オブジェクトのパスを指定し，初期の位置，大きさなどを

指定する．その他にモデルデータ内では，モデルの視点を表すカメラ座標や影の描画の有無，ライティングの有無，インタラクションの補助パラメータなども指定することができる．3D オブジェクトには一般的なモデリングソフトウェアにて制作，エクスポート可能な X フォーマットを利用し，頂点座標，テクスチャ座標を描画処理に用いる (図 3)．

```
<model name="kuma" asmpath="ModelController.dll"
                   asmtype="MC.Controller01>

  <graphics option="shadow,pick_bright">
    <x path="kuma.x" />
    <scale value="0.5" />
    <translation x="0.5" y="0.0" z="-0.5" />
    <camera>
      <position x="0.0" y="0.7" z="0.0" />
      <target x="0.0" y="0.7" z="-0.1" />
      <fov degree="60" />
    </camera>
  </graphics>

  <parameters>
    <hop flag="true" height="0.3f" />
    <move step="20" />
  </parameters>

</model>
```

図 2: モデルデータの例 (一部)

図 3: モデルの 3D オブジェクトの例

3.3 部屋の時間進行と描画

システム内ではワールド，レンダラの 2 つのスレッドが非同期に動作し，ワールドスレッドはシーン内の時間進行を管理し，レンダラスレッドはシーンのレンダリング処理を管理している．

ワールドスレッドではステップ毎にキャラクタのインタラクションを処理し，レンダラはその結果をもとにシーンのグラフィックスをレンダリングする．ワールドは 1 秒間に 20 回状態を更新し，システムが高負荷の時にも低負荷の時にもシステム内の時間進行が一定となるように実行速度を微調整しながら動作する．20 という値はスムースなアニメーションを表示するための最低限の値であると考え設定したが，この値はユーザが実行環境に合わせて変更することも可能である．レンダラはそれ以上の頻度で更新され，可能な限り高速に描画を繰り返す．

3.4 グラフィックスの生成と描画

IDHS はシーンとモデルの XML ドキュメントと 3D オブジェクトデータとを元にシーンの 3D グラフィックスデータを構築する．床と壁の 3D オブジェクトはシーンの XML ドキュメント内の数値配列を解析し頂点バッファとして生成する．またモデルの 3D オブジェクトはモデルの XML ドキュメントに指定されたパスから X フォーマットの 3D オブジェクトデータを読み込み，位置補正，スケーリングを行って生成し，シーンに配置する．シーンの初期視点はシーンにて定義されたカメラの座標から生成するが，システムの動作中にモデルの視点に任意に切り替えることも可能である．

描画システムは 1 フレームにおいて，まず背景を描画し，次に床，壁，モデルを描画し，最後に表示する文字列があれば描画する．インタラクションが組み込まれたモデルは位置や形状が変化するので随時，更新情報をモデルの描画処理に適用する (図 4)．

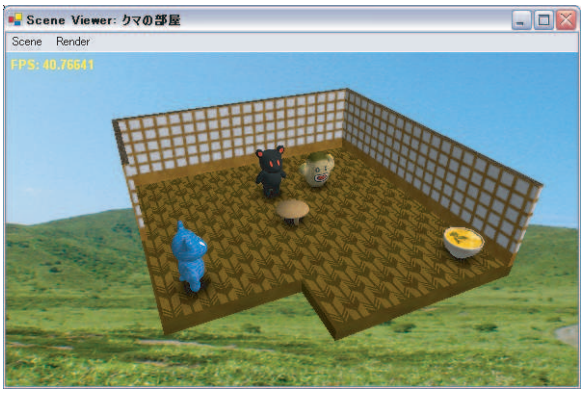

図 4: シーンのレンダリングイメージ

3.5 インタラクション

モデルのインタラクションは C# のクラスライブラリとして作成し，個々のモデルの XML ドキュメントにパスを指定することによってモデルに組み込むことが可能である．モデルへの命令は C# のメソッドとして与えられ，命令の種類はおおまかに情報型，行動型，イベント型，表現型，環境型の 5 種類に分別される (表 1)．

情報型はシーン内の地図情報やモデルの情報を取得する命令であり，行動型はモデルの移動やメッセージの発信，他モデルとの接触を行うための命令である．イベント型命令は他モデルからの命令やシステムからの命令を受信するための受動的な命令であり，シーンの更新情報，他モデルからのメッセージ，接触情報，マウス情報やキーボード情報などを受信する命令がある．イベント型は他の命令型とは異なり，メソッドの内部に処理を記述する．表現型はモデルのビジュアルを変化させるための命令であり，これを用いるとモデルの左右に揺らしたり，ジャンプさせるといった簡単なアニメーションを表現することができる．環境型は主にシステムに対しての命令であり，視点をシーン全体を見下ろすグローバル視点から，モデル自身のローカル視点に切り替える命令などがある．これらの命令を組み合わせることによってインタラクションを構成することができる．

また，インタラクションは C# のソースコードとして記述するため，IDHS にて用意された命令以外にも.NET Framework のクラスライブラリを利用できる．これを用いて，モデルをコントロールするためのフォームなどを作成することができる．

インタラクションをプログラミングするにはまず，IDHS のクラスライブラリを参照し，ModelController クラスを継承したクラスを作成する．ModelController クラスはイベント型命令を指す空のメソッドを所持しているので，必要に応じてメソッドをオーバーライドして内部に処理を記述すると，実行時にそれらのメソッドがシステムから呼び出される．また，モデルへの命令はモデルのシステム内での実体を指す Model クラスのメソッドが対応しており，ModelController クラスが所持する Model クラスへの参照である Model プロパティを呼び出し，これを用いて必要なメソッドを呼び出す．

図 5 はインタラクション記述の例である．これはコンピュータゲームの世界でよく用いられる典型的な Non Player Character(NPC) の例である．NPC とは仮想の世界での自律したキャラクタのことを指す．このモデルはランダムに全身，方向転換を繰り返し，まれに自分の名前を名乗る．そして自分以外のモデルからのメッセージを受信すると赤くなり，マウスでクリックされると大きくなるように設定されている．

このプログラムではモデルの初期化を行うメソッドである Init メソッドと，シーンが更新されると呼び出される Refresh メソッド，メッセージを受信すると呼び出される HearEvent メソッド，マウスでクリックされると呼び出される PickEvent メソッドとをオーバーライドして，それぞれの内部に処理を記述している．そしてモデルへ命令するために，Model プロパティから前に進む命令である Go メソッドや，ジャンプする命令である Jump メソッドなどを呼び出している．

モデルのインタラクションはこのようなプログラムにて構成することが可能である．

4 コンテンツの制作事例

本節ではまず，IDHS を用いた部屋型コンテンツの制作工程について解説し，それからコンテンツの作例について紹介する．

4.1 コンテンツの制作工程

IDHS のメインシステムはシーンデータを読み込み動作させる機能しか持っていない. そのため，シーンデータは別のツールを用いて制作しなければならない.IDHS にて動作可能なコンテンツの制作工程を以下に示す.

1. モデルの 3D オブジェクトと床や壁のテクスチャ画像を作成する
2. モデルのインタラクションのためのクラスライブラリを作成する
3. モデルの XML ドキュメントを作成し，3D オブジェクトのパスやアセンブリのパスを記述する
4. シーンの XML ドキュメントを作成し，床や壁のレイアウト，モデルの配置情報などを記述する

```
Information Type
        - ModelInfo GetModelInfo()
                自分のモデル情報を取得する
        - ModelInfo GetFrontModelInfo()
                自分の正面にあるモデル情報を取得する
        - CellInfo GetCellInfo()
                自分の立つ位置のセル情報を取得する
        - CellInfo GetCellInfo( int u, int v)
                指定した座標にあるセル情報を取得する
        - CellInfo[] GetAllCellInfo()
                シーン内の全てのセル情報を取得する
Action Type
        - bool Go()
                前進する
        - bool Back()
                後退する
        - bool TurnLeft()
                左に曲がる
        - bool TurnRight()
                右に曲がる
        - bool ShiftLeft()
                左に平行移動する
        - bool ShiftRight()
                右に平行移動する
        - bool Touch()
                正面にモデルがあれば触れる
        - bool Say( string message)
                メッセージを発進する
Event Type
        - void Refresh()
                シーンが更新されると呼び出される
        - void TouchEvent( int direction, ModelInfo modelInfo)
                触れられると呼び出される
        - void HearEvent( string message, ModelInfo modelInfo)
                メッセージを受信する
        - void PickEvent()
                マウスでクリックされると呼び出される
        - void KeyEvent(KeyEventArgs e)
                キーボードが押されると呼び出される
Expression Type
        - bool Jump( int time, float height)
                垂直に飛ぶ
        - bool Color( int time, int color)
                色を変える
        - bool Tilt( int time, int direction, int degree)
                傾く
        - bool Scale( int time, float scale)
                大きさを変える
Environment Type
        - bool SetSight( bool flag)
                モデルのローカル視点に切り替える
```

表 1: 命令の一部

まず，最初の工程では，既製のモデリングソフトウェアを用いて 3D オブジェクトを作成し，床や壁のテクスチャはペイントソフトウェアなどで作成する．次に，作成した 3D オブジェクトを動作させるためのクラスライブラリを Visual Studio などの開発環境を用いて作成する．単に 3D オブジェクトを表示するのみであればこの工程は必要ない．次の工程では，モデルデータを定義する XML ドキュメントをテキストエディタなどを用いて作成し，その中に 3D オブジェクトへのパスとクラスライブラリへのパスを記述する．この工程にてモデルデータが完成する．最後にシーンデータを定義する XML ドキュメントを作成し，床や壁，モデルなどの配置を記述して，コンテンツが完成する．

```
using System;
using DH.DHControl;
namespace MC {
  public class NPC01: DH.DHControl.ModelController {
    private Random rand; // 乱数を使う
    // 初期化
    public override void Init() {
      rand = new Random(); // 乱数の初期化
    }
    // モデルの更新
    public override void Refresh() {
      if( rand.NextDouble() > 0.99)
        Model.Go(); // 前に一歩進む
      if( rand.NextDouble() > 0.99)
        Model.TurnLeft(); // 左に曲がる
      if( rand.NextDouble() > 0.99)
        Model.TurnRight(); // 右に曲がる
      if( rand.NextDouble() > 0.999) {
        ModelInfo mi = Model.GetModelInfo(); // 自分の情報を取得
        Model.Say( mi.uniqueName + "だよ〜"); // 自分の名前を言う
      }
    }
    // メッセージの受信
    public override void HearEvent(string message, ModelInfo modelInfo){
      Model.Color( 20, System.Drawing.Color.Red.ToArgb()); // 赤くなる
    }
    // マウスでクリックされた
    public override void PickEvent() {
      Model.Scale( 20, 1.5f); // 大きくなる
    }
  }
}
```

図 5: インタラクションの記述の例

一度制作したモデルデータや 3D オブジェクト，テクスチャ，クラスライブラリなどのリソースは他のシーンにて再利用することが可能であるため，既存のリソースを用いる場合は 1〜3 の工程が大幅に省かれる．

4.2 迷路ゲーム

IDHS を用いた部屋型コンテンツの一例として迷路ゲームを制作した．図 6 上に示すスクリーンショットは迷路内にいるクマの「クマ夫 (図 6:左下の右側)」を操作して，ゴールにいる「クマ子 (図 6:左下の左側)」の元へ向かう簡単なゲームである．ユーザはカーソルキーを押すことによって迷路内のクマ夫を移動させることができる．ユーザはクマ夫を操作してゴールであるクマ子の元へ向かわせる．クマ子は自分の前にクマ夫が現れると「おめでとう」と話しかけ，クマ夫はそれを聞くと「やったー！」と言い，飛び跳ねて喜ぶ．また，ゲーム中にクマ夫をマウスでクリックするとクマ夫の視点に切り替わり (図 6 右下)，その視点から画面内の任意の場所をクリックすると再び迷路全体を見渡す始点に戻すことができる．このクマ夫とクマ子の実際のプログラムは図 7 のようになっており，短いプログラムによってゲームが構成されている．

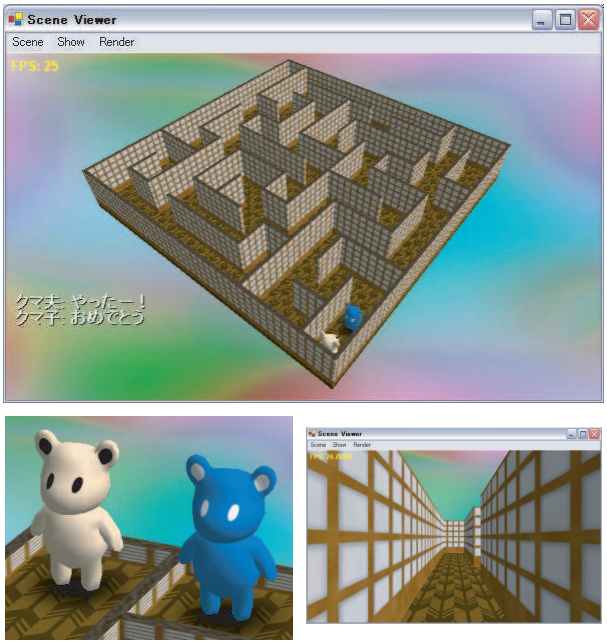

図 6: 迷路ゲーム

クマ夫のプログラム

```
using System;
using System.Windows.Forms;
using DH.DHControl;
namespace Meirokuma {
  public class MayouKuma: ModelController {
    // キーが押されたら呼び出される
    public override void KeyDownEvent( KeyEventArgs e) {
      if(e.KeyCode == Keys.Up) // ↑キーで前に進む
        Model.Go();
      else if(e.KeyCode == Keys.Down) // ↓キーで後退する
        Model.Back();
      else if(e.KeyCode == Keys.Left) // ←キーで左に曲がる
        Model.TurnLeft();
      else if(e.KeyCode == Keys.Right) // →キーで右に曲がる
        Model.TurnRight();
    }
    // メッセージを受信したら呼び出される
    public override void HearEvent(string message) {
      // 「おめでとう」と言われたら「やったー！」と言ってジャンプする
      if( message.Equals("おめでとう"))
        Model.Say("やったー！");
        Model.Jump( 20, 1.0f);
    }
    // マウスでモデルが選択されたら呼び出される
    public override void PickEvent() {
      Model.SetSight(true); // モデルの視点に切り替える
    }
  }
}
```

クマ子のプログラム

```
using System;
using DH.DHControl;
namespace Meirokuma {
  public class GoalKuma: ModelController {
    //「おめでとう」と1回だけ言うためのフラグ
    bool omedeto = false;
    // シーンの更新毎に呼び出される
    public override void Refresh() {
      // 自分の前にモデルが来たら「おめでとう」と1回だけ言う
      ModelInfo mi = Model.GetFrontModelInfo();
      if( mi != null && !omedeto) {
        Model.Say("おめでとう");
        omedeto = true;
      }
    }
  }
}
```

図 7: 迷路ゲームのプログラム

4.3 函館市の観光情報コンテンツ

図 8 に示すスクリーンショットは IDHS にて制作した函館市の観光情報を紹介するためのコンテンツである．床面のフィールドは函館市の地図を模しており，元町地区，五稜郭タワー，公立はこだて未来大学のモデルが，実際の位置に近い場所に配置されている．それぞれのモデルをマウスでクリックすると，その場所の観光情報を紹介するフォームが開く (図 8:右)．このフォームには実際の建物の写真と簡単な紹介文が掲載されており，その地区の Web ページを開くためのボタンも配置されている．これらの仕組みは.NET Framework のクラスライブラリを用いたものであり，フォームがモデルのプログラムと連携して開くようになっている．ここでのモデルはフォームを開くためのショートカットアイコンとして機能している．

また，フィールド上を自由に動き回っているクマのモデルをクリックするとクマの視線からフィールドを眺めることが可能であり，迷路ゲームと同じくカーソルキーで前後左右に移動することができる (図 8:下)．

5 システムの特徴

IDHS は多種多様の目的を持った自由な形状の部屋型コンテンツを，簡単に制作し動作させるための支援環境となっている．

ドキュメント内にて壁や床，モデルなどを定義することにより，シーンを自由にレイアウトすることができる．そして配置されたモデルには自動で移動範囲や行動ロジックなどの動作ルールが適用される．モデルのインタラクションは，その動作ルールを元にプログラミングすることが可能となっている．

モデルが実行可能な命令には，シーン内を移動する命令や，モデルにシンプルなアニメーションをさせる命令，他のモデルにメッセージを送る命令などがあり，これらを組み合わせて用いることによって複雑なインタラクションを構成できる．

IDHS では人型のモデルや家具のモデルを区別する

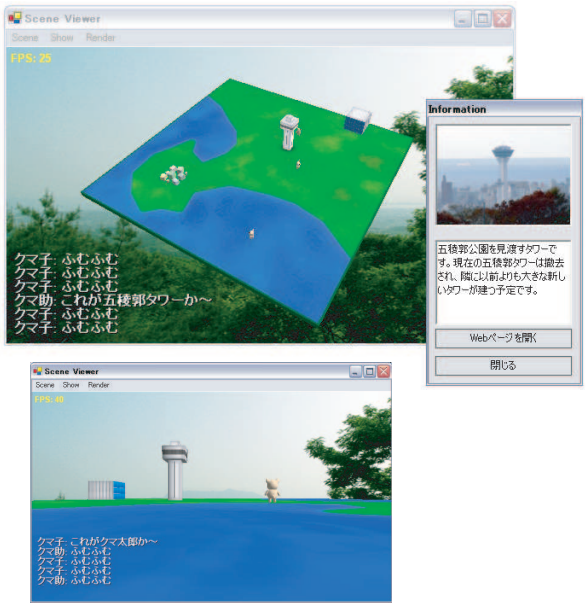

図 8: 函館市の観光情報コンテンツ

ことなく全て同じデータ構造で扱っているという特徴がある. これによって人対人のインタラクションに限らず，人対物とのインタラクションも同じルールにて作成することができる. 一方でこの特徴のために，モデルに座る，寝る，手を動かすなどの人に特有のアクションをさせることができないという弊害もあるが，それを補うために IDHS では擬人化された動作を実現するためのプリミティブなパラメータや命令が数種類用意されている.

また，モデルのインタラクションは C# によって実装するため，.NET Framework のクラスライブラリを利用して 4-2 にて説明したように，モデルにアプリケーション的な役割を持たすことも可能である. これを応用すると例えば，システムから時刻情報を取得してクリックされると現在時刻を言う「タイムキーパーモデル」や，クリックすると別のアプリケーションを開く「ショートカットモデル」，RSS リーダを実装して最新のニュースを報告する「ニュースキャスターモデル」，CPU 負荷をモニタして負荷が高くなると真っ赤になって暴れだす「CPU なりきりモデル」など，エンタテインメント性と実用性とを兼ね備えたモデル型のアプリケーションを制作することもできる.

さらに，モデルはグラフィックスとインタラクションのデータとが分離されているため，インタラクションのデータへのパスを書き替えるだけで，同じ外観のモデルに違った振る舞いをさせることができる. 同様の要領で，異なる外観のモデルに似たような挙動を持たせることも簡単にできるので，コンテンツ素材の再利用性が高い.

IDHS を用いて制作可能なコンテンツには，4-1，4-2 にて紹介したようなゲームや情報コンテンツなどがある．例えばゲームでは，モデルに成長パラメータを持たせるようにプログラミングするとキャラクタの育成ゲームとなり，壁と床をレイアウトして迷路を作り，モデルを手動で移動できるようにプログラミングすると迷路ゲームとすることもできる．また情報コンテンツでは，部屋に彫刻や絵画のモデルを配置し，仮想の美術館を制作したり，地図を模したフィールドに観光名所のモデルを配置し，モデルがマウスで選択されると詳細な情報が表示されるといった 3D の観光コンテンツを制作することも可能である.

本システムは既存のリソースを用いてシーンをレイアウトするのみであれば，プログラミングの知識は必要なく，またモデルに複雑な動作をさせたければ，プログラミング言語の機能や既存のクラスライブラリを全て利用することが可能であり，初心者から上級者まで幅広いユーザが利用できるようになっている．さらにグラフィカルなプログラムを簡単に作成することができるので，プログラミングの初心者にとっての教育的効果もあることが期待される.

6 関連研究

IDHS は部屋型コンテンツを制作し動作させるためのシステムであり，ビューワ，ツール，ミドルウェアの 3 つの要素を合わせ持つ.IDHS と部屋型コンテンツを利用，制作可能な既存のゲームや Web サービス，ツール，ミドルウェア，言語などとを，カスタマイズ性と制作の難易度とで比較したグラフを図 9 に示す.IDHS は部屋型コンテンツの制作に特化したシステムであり，汎用性の高いミドルウェアほどの自由度はないが，特定の部屋型コンテンツを制作するには高機能のミドルウェアを利用するほどの難易度はない. また，部屋をカスタマイズ可能なゲームや Web サービスと比較してコンテンツ制作の難易度は高いが自由度は高く，幅広い応用が可能となる.

関連研究には「ゲームや e-Learning インタラクティブコンテンツの制作支援用オーサリングシステム [3]」や「Lamp [4]」などがある.

前者は 3D のインタラクティブコンテンツを制作するためのシステムであるが，独自のスクリプトを習得する必要があったり，3D の座標空間を意識して設計せねばならなかったりと，利用するにはやや困難な点がある. コンパイルを必要としない環境を構築したことは評価できるが，コンテンツのカスタマイズ性がスクリプトの機能に依存するデメリットもある. また座標空間に関して比較すると，IDHS の座標空間は 2 次元のグリッド状であるためにモデルの自由な配置や移動は制限されるものの，初心者にとってレイアウトや動作アルゴリズムを考えやすいという利点がある.

一方で後者の Lamp は部屋型コンテンツに限らず，3D ゲーム全般を開発するための教育的ミドルウェアであり，一概に本システムと比較することはできないが，Lamp は高度なグラフィックスプログラミングの習得を目的としたミドルウェアであるため初心者にとっての敷居は高く，IDHS はグラフィックプログラミングを必要としないために簡単に習得できるという利点がある．両者の役割を比較すると，Lamp はグラフィックスプログラミングの学習のためのミドルウェアであり，IDHS はインタラクションのプログラミングを学習するためのミドルウェアであるともいえる.

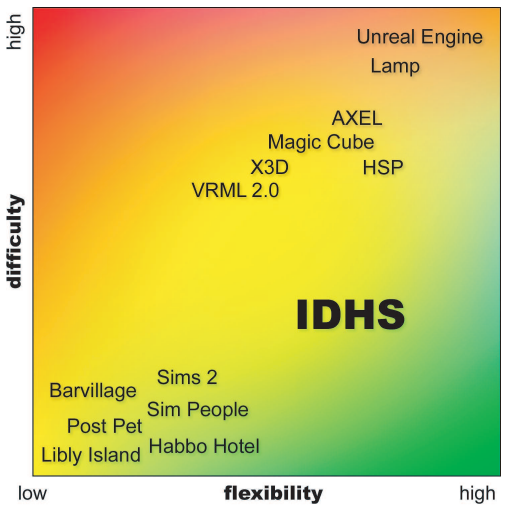

図 9: システムの位置付け

7 今後の展開

現在の IDHS ではシーンのレイアウトやパラメータの設定を行うのに XML ドキュメントを直接テキストエディタ等で書き換える必要があるが，この作業を完全に GUI 環境で行うようにすれば，初心者にとっての利便性が増すことが考えられる. またインタラクションをプログラミングせずに記述可能なツールを開発し，プログラミングが苦手なユーザでもインタラクションをデザインできるような仕組みも検討中である. さらに IDHS をネットワークに対応させることも検討中であり，これが実現されれば一つの部屋を複数のユーザが共有することが可能となり，コミュニケーションシステムとしても利用できることが期待される. この他にも，複雑なアニメーションへの対応や，グリッドに拘束されない空間への変更も検討中であるが，これらの機能はコンテンツ制作者にとっての負担を増やす恐れもあり，一概に機能拡張がシステムの改良に結びつくとは言い難い.

8 終わりに

本研究では，汎用性の高い部屋型コンテンツを制作し動作させるためのシステムを開発した. これによって様々な用途の部屋型コンテンツを簡単に制作，利用することが可能となると期待される. 本システムの評価には多くのユーザの応用事例を待たねばならないが，今後，このシステムが多くのユーザに利用されることを期待し，さらなる改良と普及に努めたい.

参考文献

[1] Electronic Arts: Sim People (2000).

[2] Sulake Corporation: Habbo Hotel (2000).

[3] 宮崎光二, 中津良平: ゲームや e-Learning インタラクティブコンテンツの制作支援用オーサリングシステムの構成, エンタテインメントコンピューティングワークショップ, pp.89-94 (2005).

[4] 大谷淳平: Lamp: 教育的かつ実用性のあるゲームミドルウェア, エンタテインメントコンピューティングワークショップ, pp.167-171 (2004).