情報教育における情報技術の習得を目的とした3Dゲーム開発手法の検討

岡崎　友佳　　高田　栞奈

要約

近年，情報技術の発展に伴って，中等教育を中心に情報教育の導入が積極的に行われている．また初等教育においても，コンピュータの利用方法を中心に情報教育の導入がなされはじめている．しかしながら，この種の授業の多くは，あくまでも情報知識の習得あるいはアプリケーションの技能の習得に限られており，情報を活用する上でのより実践的な情報技術を習得するための環境は整っていない．そこで本研究では，情報授業における情報技術の習得を目的として，これに適したソフトウェアの開発手法を提案する．開発対象は，学生の身近なソフトウェアであるゲームソフトからチェスを選び，開発を通じてソフトウェアの動作とその仕組みを習得することができる．特に本手法では，授業への導入コストや学生及び教員の理解度を考慮して，無償かつ簡便な開発手法について検討している．

1.　まえがき

近年，情報技術の発展に伴って情報のマルチメディア化が進んでおり，身近なあらゆるものに情報技術が用いられている．このため，現代社会において情報に関する知識・技術はより一般的になりつつある．特に，2003年以降，中等教育において「情報」が必修科目としてカリキュラムに導入されており，情報に対する重要性が分かる．しかしながら，この種の授業の多くは，あくまでも情報知識の習得に留まり，どのように情報を処理するかといった，より実践的な情報技術に関する内容は実施されていない状況にある．したがって，知識としては知っているがその扱い方を知らないために，情報に対する意識の剥離化が進んでおり，情報に関して誤った扱い方をしている場合も少なくない．

そこで本研究では，情報授業における情報技術の修得を目指した一カリキュラムを提案する．具体的には，ソフトウェアの開発を通じて，情報処理の際に一般的に利用されるソフトウェアの動作とその仕組みを理解する．また，開発するソフトウェアは，学生にとって身近な題材であるゲームソフトからチェスを選んでいる．以上より本研究では，チェスのゲームソフト(以降，単にチェスソフトという)の開発を通じた情報技術の修得を目的として，情報授業に導入可能な開発手法について検討する．

以降，2章において情報授業を考慮したチェスソフトの開発手法を提案する．3章では，チェスソフトを開発する上での具体的なデータ構造とアルゴリズムを述べる．そして4章で開発結果を示すと共に，その機能について説明する．

2. チェスソフトの開発手法

チェスソフトを開発する上では，ソフトウェア開発の基本となるプログラミング技術に加えてゲーム内の視覚オブジェクトに関する知識が必要となる．このため，従来この種のソフトは，高度な専門知識を有するクリエイタにより開発されている．一方，最近では，開発効率の向上を図るため，より汎用性の高い開発環境が無償で提供され始めている．この結果，携帯アプリに代表されるように，高度な専門知識を持たずとも手軽にゲーム開発が可能となりつつある．

そこで本章では，学生の基礎知識及び修得時間の観点から，情報授業で実施可能なゲーム開発手法を検討する．

2.1. プログラミング環境

すべてのソフトウェアは，プログラムに基づいて処理を行っており，これを記述することを一般的にプログラミングという．プログラミングを行うためには，開発環境が必要であり，最近では，機能制限等はあるものの，その多くが無償で提供されている．特に，ゲーム開発においては，以下の開発環境が一般的である．

・DirectX

1995年よりMicrosoft社が無償で提供しているゲーム・マルチメディア処理用ライブラリ(プログラム群)である．中でもDirectX Graphicsは高度な描画処理機能を備えて

おり，現在でも様々なソフトウェアに広く利用されている．このため，現在のWindowsベースのPCでは，DirectXのインストールが必要不可欠な状況にあり，これが広く普及していることが分かる．

・ XNA Game Studio

2006年にDirectXのゲームに関するライブラリを整理したものがXNA Game Studio(以下，XNAという)である．XNAは従来のDirectXの機能を備えているだけでなく，物体の衝突判定やゲームコントローラの入出力処理など，ゲームに特化した機能が予め定義されており，これを用いることで，専門知識を持たずとも容易に効率的なゲーム開発を行うことができる．さらに，XNAはマルチプラットフォームに対応しており，XNAで開発されたゲームはPC上だけでなくそのままXbox360でも動作することができる[1]．

以上述べたように，DirectXは汎用性の高い開発環境ではあるものの，グラフィック技術の発展途上期に開発されているために，近年のグラフィック技術の向上に伴って度重なる仕様の変更や機能拡張が行われている．そのため現在では，DirectXは非常に複雑なライブラリであり，これに基づいたゲーム開発を行うためには特に高度な専門知識が必要とされる．このため，学生の理解，修得期間を勘案すると中等教育においてDirectXを実施することは困難である．一方，XNAでは多くの機能が簡便化されているために，上述した問題にも対応することができると考えられる．したがって本研究では，チェスソフトの開発環境としてXNAを採用する．

2.2. オブジェクトの生成

近年，多くのゲームソフトウェアでは，ゲーム性を向上させるために高度な3Dグラフィックが使用されている．従来，この種の3Dグラフィックを実現するためには，開発環境内でポリゴンを一枚一枚描画していたが，グラフィック技術の高度化に伴ってポリゴン数が膨大となり，ゲーム自体の動作を圧迫することとなる．そこで最近では，3Dグラフィックに利用するキャラクター等のオブジェクトは，予めモデリングツールにより作成しておき，これをプログラム上で読み込むことによりゲームを実現する手法が一般的である．

AutoCADに代表されるモデリングツールの多くは，様々な形状を3Dグラフィックとして表現可能であり，多少の慣れは必要であるものの，その操作は比較的容易に行うことができる．しかしながら，ツール自体が非常に高価であり，学生全員のPCへは導入し難いという問題がある．

そこで本研究では，モデリングツールの中でも無償版が提供されているメタセコイアを利用する．無償版は，有償版に比してモデリング機能が制限されているが，情報授業で利用する上では必要十分な機能を備えている．さらには，多くの専門書や文献が公表されており，効率的にモデリング技術を習得することができる[2]．

以上より，本研究では，メタセコイアを用いてチェスに必要なオブジェクトを作成し， XNAによりそれらを読み込むことでチェスソフトを開発する．

3. チェスソフトの開発

3.1. オブジェクトの作成

2.2で述べたように，ゲームの3D空間上にオブジェクトを表示するためには，これに対応した3Dモデルを作成する必要がある．ここで，チェスにおけるオブジェクトは盤面とコマに大別される．前者はマスを8×8に配置した形状であることから，単純なポリゴンを配置することで容易に実現できる．一方，後者は，3次元的な造形を含むために，ポリゴンによりすべてを表現することは困難である．したがって本研究では，コマに対してのみメタセコイアを用いたモデリングを行う．

3.1.1. 作成方針

メタセコイアを用いてモデリングを行う場合，その作成形態によって以下の2つの手法に大別される．

手法1：1つの図形から全形を造形

一つの図形を変形させることで，所望のモデルを造形する．全形を確認しながら直感的に造形することができるが，細かな造形を行うためにはある程度習熟する必要がある．また，彩色の際に全形が単色となり，部位毎に彩色するためには個々に彩色領域を定義する必要がある．

手法2：複数の図形を組み合わせて全形を構成

個々に造形されたパーツを組み合わせることで，所望のモデルを造形する．これにより，パーツそれぞれに対して細かい造形を行うことができる．また，彩色も

個々に行えることから，パーツの素材を適切に表現することができる．一方，デメリットとしては，パーツの統合を考慮して各パーツを造形する必要がある．

本研究では，既存の図形の多くを再利用可能なことから，モデリング効率に優れる手法2を採用する．

3.1.2. 作成例

コマの一つであるポーンを例に，メタセコイアを用いたモデルの作成手順を以下に説明する(図1)．

手順1.頭部の作成：球体を生成し，縮小・拡大を行うことによって頭部を造形する．

手順2.髪の作成：頭部より一回り大きい球体を生成し，不要な面を削除する．

手順3.目の作成：まず，片目を球体で生成する．次に，頭部正面に対して線対称となる位置にもう一方の目を複製する．

手順4.胴体の作成：多角柱により下部，円柱により首を作成した後，これらを統合して胴体を構成する．

手順5.腕の作成：直方体を生成し，ナイフ機能(線を引いた断面で分割)を用いて新しい断面を作成する．そして，その断面を押し出して折り曲げる．

手順6.装飾の作成：ポーンに対しては，槍を装飾として所持させる．まず，基本図形の中に三角柱が存在しないため，腕と同様に直方体を生成する．次に，ナイフを用いて新たな断面を作成し，その断面を削除することで三角柱を作成する．

手順7.各部の統合：手順1～6で作成した各部のパーツを統合し，モデルの全形を構成する．

本研究では，以上の手順に基づいてチェスに用いる6種のコマ(キング，クイーン，ナイト，ビショップ，ルーク，ポーン)をモデリングしている．作成した各コマのモデルを図2に示す．

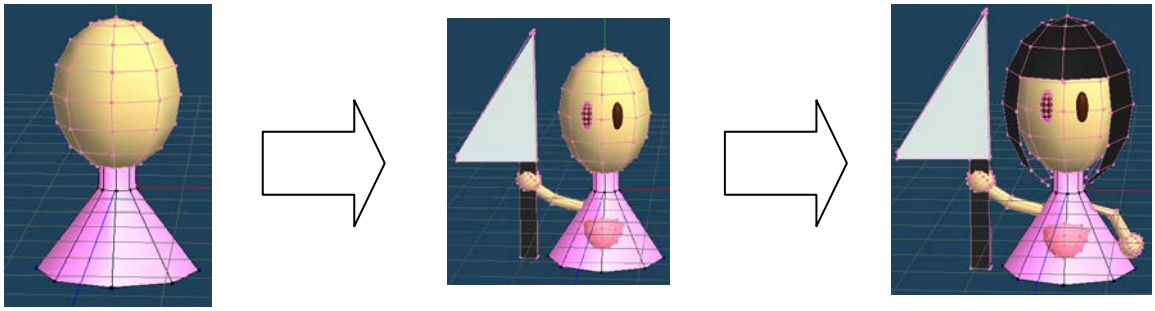

図1　ポーンのモデリング過程

キング　クイーン　ナイト　ビショップ　ルーク

図2　各コマのモデル

3.2. プログラミング

XNAによるゲーム開発において，ゲーム内の動作は，GameMainクラス(プログラムの処理体系)に定義されている以下の3つのメソッド(機能単位の処理のまとまり)により処理される．各メソッドの説明を以下に記す．

① GameMain：GameMainクラスのコンストラクタであり，プログラム上で使用するデータ構造を定義し、その初期化を行う．

② Update：0.1秒毎に呼び出されるメソッドであり，その時々におけるキーの入力状態に応じて，物理判定やオブジェクトの移動を行う．

③ Draw：3D空間上のオブジェクトを再描画するメソッドであり，一般的にUpdateメソッドに続けて呼び出される．

以降，GameMain クラスのデータ構造と各メソッドにおける処理について説明する．

3.2.1. データ構造

チェスは盤面のコマを互いに取り合うゲームであることから，チェスソフトを開発する上では盤面とコマの対応付け，つまり盤面の管理が重要となる．そこで本研究では，チェスの盤面管理に関するデータ構造として，以下の２つの手法について検討している．

手法１：盤面に対応した二次元配列を使用

盤面(8×8)に対応する二次元配列を設け，各配列要素に対して3.1により生成されたオブジェクトを格納することで，コマと盤面の位置情報を一対一に対応付ける．盤面とコマに対する処理を直観的かつ容易に行うことができるが，コマが存在していないマスにもデータ構造が存在するため所要メモリの一部が無駄となる．

手法２：コマの位置座標に対応する一次元配列を使用

コマの座標を一次元配列により管理する．一次元配列のサイズはコマ数のみであるため，手法１のように所要メモリ量に無駄がない．しかしながら，位置座標のみを用いてコマの動作を処理する必要があるため，手法１に比して処理が複雑化することとなる．

以上より，手法１と手法２では，処理と所要メモリ量に関してトレードオフの関係にある．ここで，手法１における実際の所要メモリ量は，盤面に対応して一定であるため，特に所要メモリが問題となることはない．したがって本研究では，情報授業において学生が直観的に理解し易いことから手法２を採用する．

3.2.2. コマの移動処理

3.2.1 よりコマを二次元配列により管理しているため，図3に示すように配列要素を交換することでコマの移動処理を実現できる．

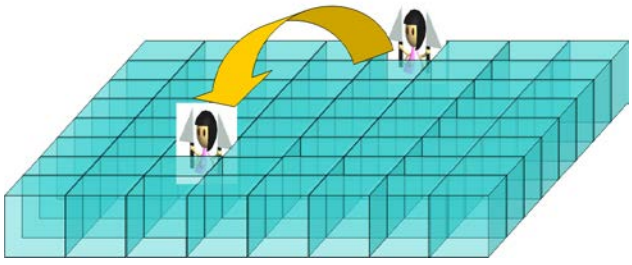

図3. コマの移動処理

ここで，コマの移動処理を行う際には，移動元と移動先に関する盤面上の移動位置を管理する必要がある．そこで本研究では，ユーザが盤面の位置を指定するためのカーソルを設け，コマと同様に二次元配列を用いてその位置を管理する．つまり，GameMain において，カーソルに対応する二次元配列 Cursor を定義する．

カーソルの移動は，Update メソッドにおける十字キーの入力状態によって行う．この時，現在のカーソルの座標を(i, j)とすると，各キーのカーソル移動処理に関するプログラムは次の通りである．

```
→キー：  Cursor[i + 1, j] = Cursor[i, j];
←キー：  Cursor[i - 1, j] = Cursor[i, j];
↓キー：  Cursor[i, j + 1] = Cursor[i, j];
↑キー：  Cursor[i, j - 1] = Cursor[i, j];
```

次に，コマを移動するためには，移動時の始点と終点の座標が必要となる．特に，終点はコマの移動処理と同時に扱うことができるが，始点は移動処理が実行されるまで保持しておく必要がある．これらの座標は，ユーザにより指定されたカーソル位置が対応するため，始点指定時におけるカーソル位置を保持するための変数 move_i 及び move_j を Game main に定義する．

そして，Update メソッドにおいて，コマの移動処理に関する２つのメソッドを定義する．

GetKoma：コマの始点座標を取得するためのメソッドである．キーボードの A が入力されている場合，カーソルの位置情報に基づいて，盤面上に有効なコマが存在するか否かを判定する．そして，コマが存在する場合は，カーソル位置をそれぞれ move_i 及び move_j に記憶した後，コマの移動状態に移行する．

SetKoma：コマの終点座標を取得し，コマの移動を行うメソッドである．コマの移動状態時にキーボードの S が押されている場合，現在のカーソル位置と move_i 及び move_j に基づいて，始点座標に存在するコマを終点座標の要素に代入することでコマを移動することができる．この時，移動先にコマが存在する場合，そのコマは代入により上書きされるため，盤面上から排除されることとなる．

以上より，これらコマの移動に関するメソッドをプレイヤー1 と 2 それぞれ交互に処理することで，チェスの基本的な動作を実現できる．

3.2.3. 移動可能マスの強調処理

一般的なチェスソフトでは，コマを選択した際に移動可能なマスが強調して表示される．そこで，本研究では，これを実現するため，移動可能場所の設定メソッド PlaceSet を定義する．また，移動可能なマスは，カーソルと同様に盤面に対応した二次元配列 Place により管理するものとする．したがって，PlaceSet で行われる処理は，選択中のコマが移動可能なマスに対応した Place の要素に(ポリゴンにより生成された)パネルオブジェクトを割り当てることである．

例として，ルークの移動可能マスの強調処理について述べる．ルークは縦横かつ無制限に移動可能であり，その移動範囲は，現在位置に依って変化する．そこで本研究では，for 文を用いて現在位置に対して柔軟にその移動可能マスを設定する．ルークに対する移動可能マスの強調処理プログラムは次の通りである．

```
for(int x = 0; x < Width; x++){
  for(int y = 0; y < Height; y++){
    if(i == x || j == y)Place[x, y] = panel;
  }
}
```

ここで，Width 及び Height はそれぞれ盤面の幅及び高さであり，panel はパネルオブジェクトである．二次元配列上を走査し，コマの X 座標または Y 座標上ならばパネルオブジェクトを代入している．

同様にすべてのコマの移動範囲に対する強調処理プログラムを定義することで，選択中のコマに対応してその移動範囲を強調表示することができる．

3.2.4. カメラ処理

チェスソフトにおけるユーザインタフェースを考慮すると，盤面を 3D 空間上のどこから見るのかというカメラ視点が重要となる．例えば，視点が遠すぎると盤面上のコマに見えにくく，近すぎると盤面全体が見えないこととなる．したがって，これらカメラ視点に関しては，プレイヤーが自由に調整可能であることが望ましい．

そこで本研究では，盤面を中心とする半径 R の球体上をカメラが移動するものとする(図 4)．このため，球体上の極座標をカメラ座標(X,Y,Z)に変換する必要がある．

まず，XY 平面について考える(図 5)．このとき，X 軸と Y 軸がなす角を θ とすると，X 及び Y 座標はそれぞれ次式により与えられる．

$$X=R\cos\theta \qquad (1)$$

$$Y=R\sin\theta \qquad (2)$$

次に，XZ 平面について考える(図 6)．このとき，X 軸と Z 軸がなす角を ϕ とすると，Z 座標を次式より与えられる．

$$Z=R\cos\theta\sin\phi \qquad (3)$$

以上，式(1)～(3)より，R の値を変化させることでカメラの距離が変動し，θ と ϕ の値を変化させることでカメラの視角を変更することができる．

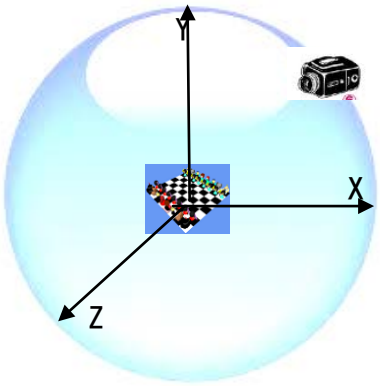

図 4. カメラ位置

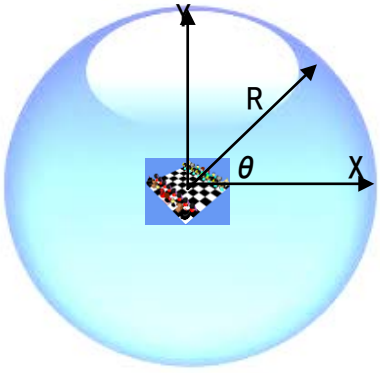

図 5. XY 平面

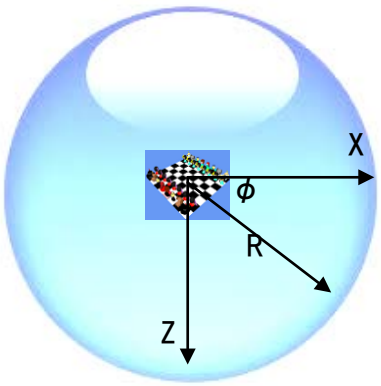

図 6. XZ 平面

4. 開発結果

3. に基づいて開発したチェスソフトの動作画面を図 7 に示し，各機能の説明を以下に記す.

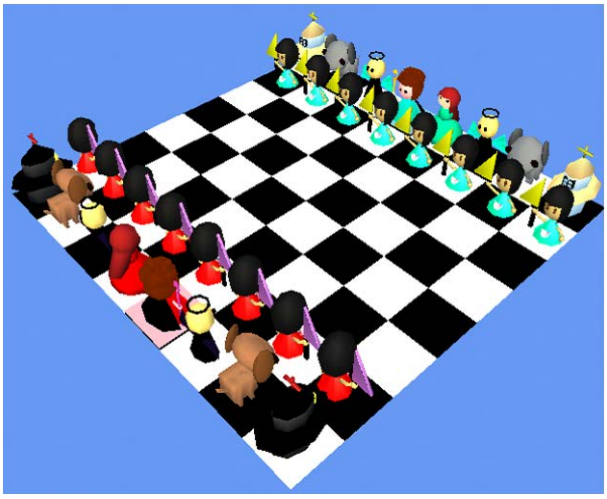

図 7　動作画面

コマの操作：カーソルは色付のパネルとして表示されており，十字キーにより盤面上を移動できる．対象とするコマにカーソルを合わせた後，キーボードの A を入力することで，そのコマを移動対象とする．このとき，コマの種類によって，移動可能なマスに色付パネルが展開される．続けてカーソルを操作し，移動先のマスを指定する．ただし，選択可能なマスは，上述した移動可能マスに限定される．そして，キーボードの S を入力することでコマを移動することができる．コマの移動を終えると，ターンが相手側に移る.

カメラの操作：シフトキー押しながら十字キー等を入力することでカメラの位置を操作することができる．このときの各キーに対応するカメラ操作は次の通りである.

→キー：θ 方向に右移動
←キー：θ 方向に左移動
↓キー：φ 方向に下移動
↑キー：φ 方向に上移動
A キー：中心方向に前進
Z キー：中心方向に後退

5. あとがき

本研究では，メタセコイアと XNA GameStudio を用いたゲーム開発手法を検討し，容易に情報技術を習得可能であることを確認した．今後の課題としては，本手法に基づいた情報教育を実施し，その有効性を確認することである.

6. 謝辞

本研究にあたり，熱心にご指導，ご助言して下さった谷野先生，ＴＡの近藤先生に心より御礼申し上げます.

7. 参考文献

[1] 赤坂 玲音：”XNA ゲームプログラミング Xbox 360 と Windows のクロスプラットフォーム開発”，ソフトバンククリエイティブ，2009

[2] 原田 大輔:”新 メタセコイアからはじめよう！ ”，技術評論社 ，2009