モーションコントローラを用いた複合現実型ゲーム開発：Microsoft Kinect によるケーススタディ

岸 将大[*1] 小元 翔太郎　坂本 瑞季　佐久間 裕之　平出 聡　野元 隆介
山邉 哲生　中島 達夫

Mixed Reality Game Development with Motion Controllers: Case Studies with The Microsoft Kinect Sensor Device

Masahiro Kishi,[*1] Shotaro Komoto, Mizuki Sakamoto, Hiroyuki Sakuma, Satoshi Hirade, Ryusuke Nomoto, Tetsuo Yamabe, Tatsuo Nakajima

Abstract –
我々は研究室配属直後の学部 4 年生を対象とした実習の中で Microsoft Kinect を利用し，二種類の複合現実型ゲームを開発した．API が提供されている廉価なモーションコントローラが普及する中で，一般の開発者が複合現実感を活用したシステムを構築することが容易になりつつある．そこで本研究ではケーススタディのデモ展示を通して，複合現実型ゲームを開発する上で考慮すべき設計要因や技術的課題を指摘する．

Keywords : Mixed reality, Microsoft Kinect, motion controllers, game design

1. はじめに

近年の家庭用ゲーム機の変革に見られるように，コントローラデバイスの代わりに手や腕のジェスチャ，また体の姿勢・動作を入力とするインタラクションスタイルが普及しつつある．十時キーやボタン操作を覚えずともプレーヤーは体を動かすことでサービスを利用することができるため，より幅広いユーザ層を対象とした”直感的”なインタラクションが可能になると期待される．このようなモーションコントローラは複合現実感の実現に欠かすことのできない機能要素であり，例えばシミュレータを用いた技術習得支援においてはコントローラデバイスを用いる必要がないため，仮想環境で習得した所作をそのまま現実世界にも適用することができる．任天堂 Wii では Wii リモコンを様々な道具（指揮棒や楽器，剣）に見立ててゲームを楽しむことができるが，デバイスの形状やインタフェースの違いから本物を扱うスキルを身につけることはできない．道具にセンサやボタンなどのインタフェースを付加したり，模擬デバイスを利用するようなアプローチは本物のデバイスとコントローラデバイスの使用感の乖離が大きいなどの問題点があった．しかしながら，環境側で行動認識が行えるようになったことで道具は従来のものを使用できるなどユーザ側の自由度が広がり，より実環境に近い状態でサービスを利用することができるようになる．

2010 年に Microsoft 社より発売された Kinect [1] はジェスチャや音声認識による入力を可能としたゲームシステムで，RGB カメラや深度センサなどを搭載している．主な用途として Microsoft Xbox 360 向けのゲーム開発に利用される一方で，公式，非公式を含め種々のミドルウェアやライブラリ群が提供されており，一般のユーザでも Kinect を利用したシステム開発が可能である．例えば，MikuMikuDance [3] や Interactive Puppet Prototype with Xbox Kinect [2] は Kinect を既存のシステムや従来のエンターティメントに適用・発展させている例である．学術的な研究用途においても，複合現実感を活用したシステムを開発する際，ユーザの行動認識を実現するためのコストを大幅に下げることができる．しかしながら，発売されたばかりであることもあって，Kinect を利用したシステム開発において考慮すべき設計要因や技術的課題については不明なところも多い．そこで我々は 2011

*1: 早稲田大学 理工学術院 基幹理工学部・基幹理工学研究科, {kishio7c, komoto, mizuki, hiroyuki, hirade, nomoto, yamabe, tatsuo}@dcl.info.waseda.ac.jp
*1: Dept. of Computer Science and Engineering, Waseda University

年の 4 月に研究室に配属されたばかりの学部 4 年生を対象にプログラミング実習を計画し，Kinect デバイスを利用したアプリケーション開発を行った．本プロジェクトは六人の学部生を三人ずつ二つのチームに分け，アプリケーションのコンセプト立案から開発まで約二ヶ月をかけて行った．2 章で脳カベ，3 章で Carrying Game それぞれのアプリケーションを説明する．また Kinect を用いた一般的な複合現実感システムの開発における課題発見だけでなく，プログラミング初学者に対する教育的側面についても考察，報告する．

2. ケーススタディ 1：脳カベ

2.1 ゲーム概要

脳カベは接近してくる壁に空いている穴の形状に体を合わせ，壁を通り抜けるというゲームである．日本のテレビ番組を始めとして，海外でも「Brain Wall」や「Human Tetris」の愛称で親しまれている．このようなアトラクションは大掛かりな仕掛けや機材が必要であるため，一般の消費者が実際に体験して楽しむことは難しい．そこで，本ケーススタディでは Kinect を用いてディスプレイ内の仮想空間で脳カベを体験することができるゲームを開発した．図 1 にゲームプレイ時の様子を示す．

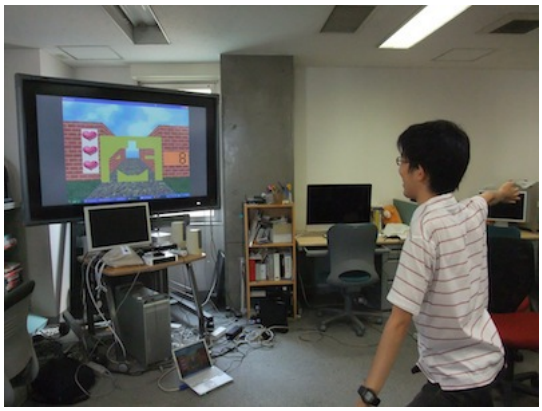

図 1　脳カベゲームプレイ時の様子
Fig. 1　Brain Wall game appearance

ゲームを起動すると，まずメニュー画面が表示され難易度を選択できる．難易度によって壁に空いている穴の形状が変わり，難易度が高い場合はプレーヤーは複雑な姿勢を一定時間維持しなくてはならない．ゲームが開始すると図 2 のようなゲーム画面に切り替わり，難易度に応じた壁が一定速度で迫ってくる．プレーヤーは壁に空いた穴に姿勢を合わせることで壁を仮想的に通過する．

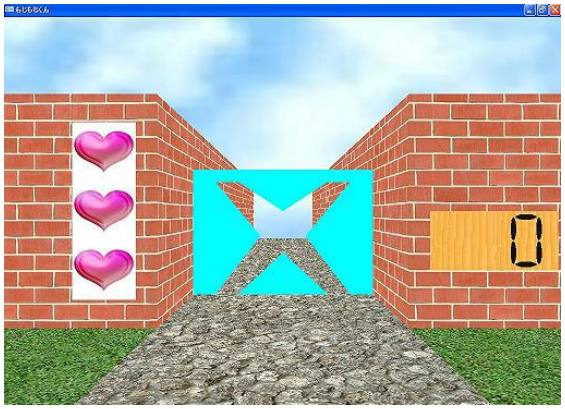

図 2　脳カベゲーム画面

Fig. 2　Brain Wall screenshot

画面には，図 3 のようなフィードバック画面に Kinect が検出した姿勢情報に応じてそのまま壁を抜けられるかどうかの判定結果が表示されるので，プレーヤーはそれを元に姿勢を補正することができる．壁にぶつかってしまうとライフポイントが減り，ポイントが全てなくなるとゲーム終了となる．

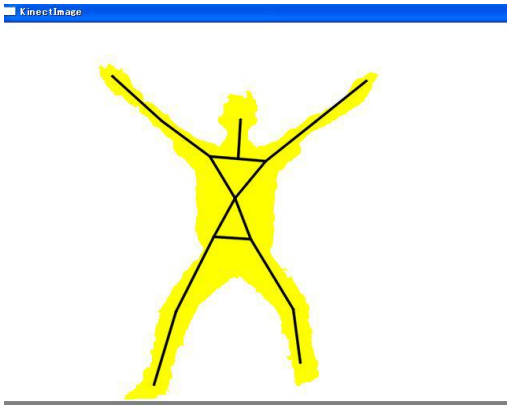

図 3　脳カベフィードバック画面

Fig. 3　Brain Wall feedback screen

2.2 システム構成

本システムの開発ではメニュー画面，ゲーム画面，Kinect からのスケルトン情報取得と通過判定の三つの部分に担当を分け，並行して開発を行った．本システムはモーションコントローラによって全てのインタラクションを行うため，メニュー画面においては右手の掌の位置にカーソルを表示し，一定時間アイコンの上に手をかざし続けると選択するという操作方法を採用した．また仮想 3D オブ

ジェクトの描画には OpenGL [5] ライブラリを使用した．プレイヤーと壁の衝突判定には OpenNI [4] ライブラリを用い，Kinect から得られたスケルトン情報から肘や股関節や頭などの体の各部分の座標を測定して壁の穴の形状のエリア内に収まっているかどうかを計算している．また現状ではフィードバック画面の生成に OpenCV [6] ライブラリを利用している．開発環境には Visual Studio 2010 を用い，C++ によってコーディングを行った．

現時点ではシングルプレイヤーでのゲームのみをサポートしているが，今後も開発を継続して複数プレイヤーに対応し，得点による勝負などを可能にする予定である．

3. ケーススタディ2：Carrying Game

3.1 ゲーム概要

二つ目のケーススタディとして，仮想的なトレーを使ってバランスを維持しながらゴールまで物を運ぶ Carrying Game を開発した．プレイヤーは強制スクロールする背景の中で自らのトレーを操作し，障害物を避けなければならない．コース上には障害物だけでなく，アイテムも設置されており，アイテムを取得することで得点が可算される．障害物に接触すると減点される他，トレーの上に乗せたオブジェクトを落としてしまうと得点が半減し，その上タイムロスとなってしまう．図 4 にゲームプレイ時の様子を示す．

図 4　Carrying Game ゲームプレイ時の様子
Fig. 4　Carrying Game appearance

プレイヤーはトレーを持ち上げたり左右に傾ける動作を行うことによって画面上のトレーの位置を操作することができる．従って脳カベとは異なり，プレーヤーは次々に現れる障害物やアイテムに応じて動的に姿勢を変えなければならない．また，仮想的なオブジェクトを使用するため両手を使って実際には存在しないトレーを保持する姿勢を保たなければならない（ただしトレーのような板を保持しても Kinect による姿勢認識は可能であるため，プレーヤーが姿勢を維持しやすいように補助的に利用することもできる）．図 5 に Carry Game のゲーム画面を示す．

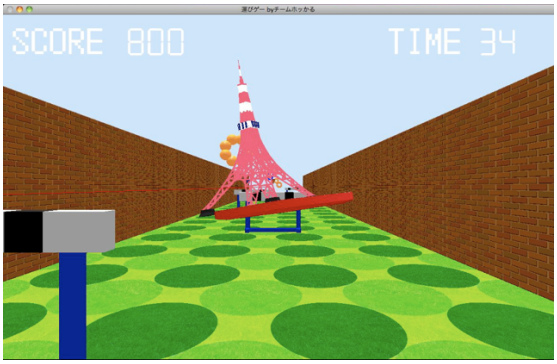

図 5　Carrying Game ゲーム画面
Fig. 5　Carrying Game screenshot

Carrying Game においても脳カベ同様，ゲーム開始時に異なる難易度のステージを選ぶことができる．また更に図 6 のオブジェクト選択画面に示されるように仮想トレーに乗せるオブジェクト（東京タワー，箒など）によって接地面積や重心が変わるため，オブジェクトの選択によっても難易度を調整することができる．Carrying Game も脳カベ同様，モーションコントローラのみを用いてのゲーム操作を提供しているため，メニュー画面での選択操作には一定時間手をかざす動作を用いた．

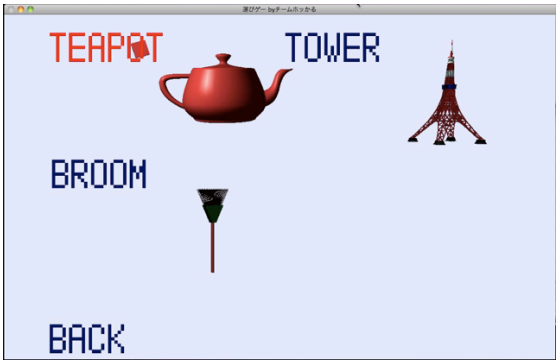

図 6　Carrying Game オブジェクト選択画面
Fig. 6　Carrying Game object menu

3.2 システム構成

本システムの開発に使用した言語は C++であり，主なライブラリとして 3D の CG 描画に OpenGL，

Kinect によるスケルトン認識に OpenNI を用いた．スケルトン認識により実際の人の手の x，y，z 座標を取得し，それらを画面内のオブジェクトの座標と対応づけることで，実世界の手の動きに連動して画面内のオブジェクトが動くようにした．また，本システムは 4 つの画面（スタート画面，ステージ選択画面，トレー上オブジェクト選択画面，ゲームプレイ画面）から構成されている．ゲームプレイ画面においては，得られた両手の座標からトレーの傾きを計算し，その傾きに応じてオブジェクトがトレー上を移動するよう，描画をおこなった．そして，ステージの床や壁にはテクスチャを張って模様を描く，SDL_mixer ライブラリを用いてゲーム中の BGM や効果音を鳴らす等，ゲームシステムの基幹となる部分以外の面の演出にもこだわった．

4. 考察

本章では，本プロジェクトにおいて Kinect を用いた複合現実感アプリケーションの開発プロセスにおいて得られた知見について報告する．

本プロジェクトで使用した Kinect は，公開されている API によってプレイヤーのスケルトン情報を座標として取得することが可能であるため，任意のタイミングでの姿勢や一定時間手を動かさず任意の位置にかざすというような比較的静的な行動認識は初学者にとっても容易に行うことできた．実際，今回ゲームを開発した本研究室の学部生はこれまでセンサや画像認識を用いた行動認識に取り組んだことはなかったが，OpenNI や OpenCV のように豊富な API やライブラリ群が提供されていたため僅か二ヶ月の期間でゲームのコンセプト企画から開発までを行うことができた．複合現実感を活用したシステム開発を行う上での敷居は着実に下がってきており，今後も様々なメディアや研究において同デバイスが活用されることが予想される．Kinect プログラミングにおける開発事例が少ないことから API やコーディング，またライブラリのインストールやパスの設定など開発環境を構築するための（特に日本語の）ドキュメントは充実しているとは言えないが，この点に関しても今後コミュニティの拡大に伴って次第に解消されていくと考える．

一方で，Kinect を用いた開発ではプレーヤーのスケルトン情報を入力としてシステムを動作させデバッグを行うことになるが，逐一 Kinect の前に立ちキャリブレーションの姿勢をとり，認識された後に全身を動かして特定の動作や姿勢を再現するのは体力と労力を必要とする作業である．GUI アプリケーションの開発では特定の順序とタイミングでキー，またはマウスイベントを発生させてデバッグを支援するようなシステムがあるが，モーションコントローラを利用するアプリケーション開発においてダミーの姿勢やモーションデータをコマンドに応じて送出するようなシステムの重要性はより高くなると言える．この際には例えば Kinect を接続しなくてもデバッグモードとしてアプリケーションが立ち上がり，GUI からの操作でダミーデータを操作しながらデバッグを行うことができるようになるのが好ましい．

モーションコントローラによるインタラクションでは，正しくセンサがユーザの位置や姿勢，行動を検知していることを適切に伝えることが必要となる．例えば OpenNI では初期化フェーズにおいて，両腕を水平に延ばした状態から肘から先を直角に上方に曲げるという姿勢をとることでキャリブレーションが完了する．しかし，今回開発したゲームにおいてはしばしば全身が映っていないなどの理由でキャリブレーションに手間取ることも少なくなかった．ユーザは Kinect に対して 1.8m 離れることが推奨されており[1]，近づき過ぎると特に膝から足先にかけてはカメラの認識範囲から外れやすい．このようなケースでは Kinect が取得しているカメラ画像などの情報を提示し，プレーヤーに適切な位置取りを促すことも必要である．

また，プレーヤーの服装や認識範囲内に椅子や棚などの障害物があると認識精度が落ちる場合がある．例えばワンピースを着た長髪の女性のようなシルエットが認識され辛かったり，至近距離にハンガーにかかった服などがあるとこれを人として認識してしまうことがある．背景がカーテンのような単色のものであると問題はないが，プレーヤーと同程度の距離に障害物があるとそれらを検出してしまうことがある．このため，開発やデモンストレーションの際には十分なスペースを確保することが課題となるが，環境によってはこのような物理的な制約を解消することが難しいケースも少なくない．また，Kinect のように全身を動かすゲームに置いて服装はゲームの操作性にも影響

1: 一人プレイ時，二人の場合は約 2.5m

を及ぼし，例えばスカートを穿いたまま大きく足を動かすことは難しい．

最後に，モニタの中に構築された仮想世界においてはプレーヤーの動きに対するフィードバックが重要である．例えば Carrying Game においてはプレーヤーのアバターは表示されないものの仮想的なトレーが代わりにプレーヤーの位置を示している．一方，現在のバージョンの脳カベでは，現状ではゲーム画面とフィードバック画面が別々に実装されているため，プレーヤーは正しい位置取りをしているかや正しい姿勢をしているかなどを直感的に判断することが難しい．プレーヤー自身の状況を判断するためにはそのアバターを見るだけで判断できることが好ましく，この点に関してはデモンストレーションまでに改良を行い，ゲーム画面の統一を行う予定である．一方，触覚や音声などの視覚とは異なるモダリティを用いたフィードバックは視覚的な認知資源の消費を抑えることができるため，プレーヤーの注意を複数のウィンドウやディスプレイに分散することなく仮想空間へ没入させることができる．現状では付加的なデバイスを用いない限り触覚に働きかけるフィードバックを与えることが難しいため，例えば壁や障害物と接触した場合では効果音を用いたフィードバックを採用しているが，仮想現実感の実現という観点におけるよりリアリティのあるフィードバックのデザインについては更なる検討が必要である．

5. 結論

本論文では，研究室配属直後の学部４年生を対象に実施したプログラミング実習において開発した複合現実型ゲームである脳カベと Carrying Game を紹介した．また Microsoft Kinect のような廉価なモーションコントローラを活用したシステム開発が今後増えていくことを想定し，本プロジェクトにおいて得られた知見を報告した．デモンストレーションでは，今回開発したゲームを参加者に体験してもらい，得られたフィードバックをもとに更なる改良を行っていく予定である．

参考文献

[1] Microsoft Kinect, http://www.xbox.com/ja-JP/kinect.

[2] Emily Gobeille and Theo Watson, Interactive Puppet Prototype with Xbox Kinect, http://vimeo.com/16985224.

[3] Vocaloid Promotion Video Project, http://www.geocities.jp/higuchuu4/.

[4] OpenNI framework, http://www.openni.org/.

[5] The Industry's Foundation for High Performance Graphics, http://www.opengl.org/.

[6] Open Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/.