Leap Motion を用いたアート制作

—— 仮想空間における物体の把持と解放を中心に ——

永江孝規
インタラクティブメディア学科

Art Production by Use of Leap Motion
— Focused on Grabbing and Releasing Objects by Hands in the Virtual Space —

NAGAE Takanori
Department of Interactive Media

(Received October 31, 2018；Accepted December 20, 2018)

Abstract

A hand gesture input device, called Leap Motion, has been utilized in virtual reality artworks since 2012 to capture hand and finger motions, with comparatively reliable precision. The device detects all the coordinates of the finger joints, as well as the positions of both palms and wrists in real time. Although many implementations have been proposed, some difficulties remain for grabbing or releasing 3D objects with the virtual fingers. Each of the fingers as well as the palms has its own collider, to detect collisions with objects, but sometimes it simply bumps against objects, pushing away them, unable to hold objects stably in the palms. I attach an inner collider and an outer trigger to an object to interact with finger-tip triggers for smoothly grabbing and releasing the object.

1. はじめに

光学式もしくは磁気式のモーションキャプチャシステムが CG 制作に応用されるようになって久しい。『トータル・リコール』（1990 年）では、Metrolight Studio 社の VFX スーパーバイザーであった Tim McGovern が、それまでゴルフスイングの分析に使われていたモーションキャプチャシステムを骸骨のアニメーションに応用した（Mocap was first being used to analyze golf swings at the time.）[1]。またセガの鈴木裕によれば、それまで医療にしか使われてこなかったモーションキャプチャを初めてビデオゲームに応用したのは 1994 年に発売された『バーチャファイター 2』であったという（At that time, motion capture technology was used only in health care, and we were the first to apply that technology in a game, which was VF2.）[2]。これと並行して、人間の視覚と知覚を模倣してステレオ映像などから人体の姿勢を推定するコンピュータビジョン的手法が研究されてきた[3) 4) 5)]。今日では Perception Neuron[6] など加速度センサーを用いた簡易なキャプチャデバイスの利用が増えており、また Wii や PlayStation、Xbox などゲーム機用に開発された比較的安価な多眼カメラデバイスによってもモーションキャプチャが実現されている。学術分野ではオプティカルフローやディープラーニングなどを用いて、単眼もしくは多眼のリアルタイム映像から 2D もしくは 3D の人体姿勢を推定するシステムの研究が進んでいる[7) 8)]。

ほんの 10 年前には、実時間で人体姿勢を検出するシステムを構築することは大学院レベルの研究課題であったが、今では Kinect に代表されるように、そうしたシステムはすでに民生用の既製品として市販され、容易に入手可能であって、誰もが手軽にパソコンに接続して利用することができる。Kinect ほど広く知られてはいないが Leap Motion もまた、極めて安価に入手できるリアルタイムのモーションキャプチャ装置である。Leap Motion を入力デバイスとして活用すれば Unity や Unreal などのゲームエンジンの可能性をさらに広げることが可能である。本稿では、主に Leap Motion とゲームエンジンを学生によるインタラクティブアート作品制作に応用することを目的として、仮想空間において精神的ストレスなく、物体をつかんだり離したりする手法を提案する。

2. Leap Motion と SDK、及び先行研究

Leap Motion は 2012 年に登場した簡易で安価な、手指の姿勢キャプチャに特化した非接触式の入力デバイス

である（図 1）。小型で軽量なために最近では HMD（ヘッドマウントディスプレイ）に装着して VR（バーチャルリアリティ）に用いられることも多い。Leap Motion は横幅 8cm、奥行き 3cm という極めて小型な筐体に 3 つの赤外線 LED と 2 つの赤外線カメラを一列に並べた装置であり、2 眼ステレオ視によって距離画像を計測するが、この手法自体決して目新しいものではない。即ちハードウェアとしての Leap Motion コントローラーは単なる距離画像計測装置というにすぎない。距離画像から人体の姿勢を推定する技術は Wii や Kinect などのデバイスにも採用されている。Wii はどちらかと言えばゲームとして成立する最低限の姿勢情報が得られればそれで良しとする。一方 Kinect の開発者たちは、単にゲームの入力デバイスとしてというよりは、汎用的な姿勢認識技術の獲得を追求している。Leap Motion は Wii や Kinect とは一線を画した独自路線を行く。ただひたすらできるだけ簡易なシステムで手指の動きをキャプチャすることをその使命としているのである。

インタラクティブアート作品を制作し展示するときに、私たちはしばしばデバイスの信頼性や安定性に悩まされることになる。設置にキャリブレーションが必要だったり、照明の影響を受けたり、デバイスドライバーがうまくインストールできなかったり、部品数が多くコネクタの接触不良等でセンサーがうまく作動しなかったり、手荒に扱うと壊れたりする。また、センサーなどのデバイスはしばしば高価であり、場所を取ったり搬送に手間取ったり保管場所に苦労したりする。しかしながら Leap Motion は単価も安く、多少のことでは壊れず、しかも非常にコンパクトであり、いざというときにうまく稼働しないなどというトラブルもほとんどない。

Leap Motion の姿勢推定アルゴリズムは明らかにされていないが、距離画像を用いた関節位置のマッチング手法であることは間違いあるまい[9) 10)]。そしてたとえ近年流行している機械学習などによる最新の推定・追跡手法を用いるとしても、遮蔽（オクルージョン）が発生するとしばしば姿勢獲得に失敗する。しかしながらデータグローブなどの体に装着する方式のモーションキャプチャ装置に比べて Leap Motion の簡便さは魅力であるし、さほど正確さを必要としないアート作品などに応用することには十分に意味がある。若干の精度向上は今後も見込めるものの、手指に何の制約も加えない Leap Motion の場合、触覚や力覚などのフィードバックを手に伝えることはできないので、Leap Motion を（ロボティクスやサイバネティクスで言うところの）正確な遠隔操作を行うためのマジックハンドやマニュピレーターの代用にすることは今後もできないし、また（石井浩のタンジブル・ビットのような）遠隔で触覚を伝えるアート作品の制作も不可能である。

Leap Motion によって獲得された指や手のひらの座標にボーンやメッシュを割り当てることで仮想の手指を表示することができる。さらにボーンにコライダー（collider; 当たり判定用の 3D 形状）を割り当てることによって、3D 物体と手指のインタラクションが、たとえば Unity や Unreal などのゲームエンジンを使いることで、比較的簡単に実現できる。しかしながら、物体をつかんだり、離したり、あるいはつかんだ状態から投げたりするためには、コライダーと物体の干渉を抑え、手のひらの中に物体を安定して保持する工夫が必要になる。

Leap Motion から得られるデータは、図 2 に示すように、50 組程度の、肘や手、指の関節位置の 3 次元座標に過ぎない。

Leap Motion を使用するにあたって Unreal と Unity 両方の SDK（開発環境）を検討したが、どちらも本質

図 1　Leap Motion の外観。テーブルの上に置かれた、マウスの左側にある装置が Leap Motion である。

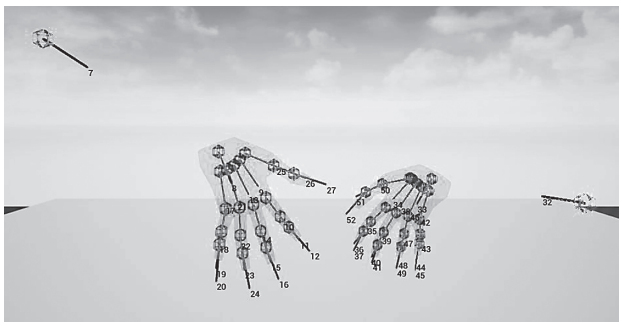

図 2　Leap Motion で測定される関節位置

的には同じである。学生は高品質な Unreal を好み、かつ手指のモーションやゼスチャーでパーティクルを発生させるなどの派手なエフェクトを使いたがる傾向がある。しかし Unreal は（高品質のシェーダーなどを用いているために）高い PC のスペックを要求し、ノート PC や非力なデスクトップ PC で気軽に作業することが困難である。今回は Leap Motion を用いて物をつかんだり投げたりできるかどうかという基礎的な検証を行いたかったために比較的 PC に負荷が少ない Unity を用いることにした。

通常、物体どうしの衝突判定（collision detection、当たり判定とも）には、物体そのものの形状ではなく、物体をコライダーで覆って、コライダーどうしが重なったかどうかを判定する。物体は通常メッシュ（ポリゴンの集まり）で表現され、メッシュは物体の形状やマテリアル、テクスチャなどを表現する。メッシュは当たり判定には向いておらず、コストがかかりすぎるので、より単純な幾何学形状（直方体、球、円筒など）をコライダーとして、それでメッシュを包み込んで、メッシュの代わりにコライダーどうしで当たり判定を行う（例えば球面と無限平面の交差は二次方程式に解が存在するかどうかで判定でき、交差箇所はその解から求まる）。物体とコライダーを別にすることによって、コライダーを物体よりもやや大きく膨らませたり（あるいは逆に小さく縮めたり）、また手指などの複雑に変形する物体の場合には、複数のコライダーを組み合わせて当たり判定を近似する。

Leap Motion から得られる手指の座標データにコライダーを割り当てて物体との当たり判定を行うだけでもある程度、手指による物体の把持や解放をシミュレートすることはできる。特に物体が無重力空間に浮遊しているだけの場合など、指で包み込むことで物体をつかんだり離したりすることができる[11]。しかしながら物体に常に力が作用している場合、たとえば、地上のように、物体が常に重力の影響をうけているときなど、コライダーによる当たり判定は不安定になり、予想外の反発力が発生する。

より一般的に言えば、多くの力が複雑に関係すればするほど、どのような反発力が発生するか予測不可能になる。より確実に物体をつかみ、離すためには、コライダーや重力などの物理エンジンを使わない方法、つまり親子関係のような階層構造を用いる必要がある。

今回の実験には Leap Motion 社が配布している Unity Assets for Leap Motion Orion Beta[12]の Core Assets 4.4.0[13]だけを用いた。このほか Unity には、2016 年頃から開発が始まった Leap Motion Interaction Engine というパッケージが用意してある[14]（2018 年 10 月現在のバージョンは 1.2.0）。この Interaction Engine とは、Interaction Behaviour.cs というスクリプトを物体に仕込んで Leap Motion コントローラーとのインタラクションを実現するというものである。かなり複雑なパッケージで、そのドキュメントには「ユーザーがいつ物体をつかもうとしたかを検出するための、うまく調節されたヒューリスティックが実装してある（we've implemented a finely-tuned heuristic for detecting when a user has intended to grasp an interaction object.）」と書かれており、「ヒューリスティック」と言うからには、正しい方法である保証はないが、ある程度のレベルまで役に立つ手法、というような意味であろう。実際、いつユーザーがどの指を使って物体をつかもうとしたか、離そうとしたか、投げようとしたかということは、Leap Motion から得られる情報だけで確実に知ることはできず、ある程度まで、手指の姿勢やモーションから推測するしかない。

Interaction Engine のソースコードの註釈を読む限りにおいて、およその原理は次のようなものであると推測される。親指とそれ以外の 1 つの指が同時に物体に触れたときに「つかんだ姿勢（grasped pose）」になったと判定し（In InteractionHand.cs: A grasp will only begin if a finger and thumb are both in contact with the interaction object.）、grasped position（おそらくは手のひらに対する物体の相対位置）を得て、手のひらと物体を連動させる。つかんだ指が物体から離れた（released）ら、物体を手のひらに追随させなくする。

このほか Leap Motion SDK for the Unreal Engine [15] 及びそのモジュール [16] が公開されている。こちらは現在のところ Unity ほど開発は進んでいないように思われる。grabAngle と pinchDistance という関数が用意されており、grabAngle は指を曲げた角度、pinchDistance は親指と人差し指の間隔を返す関数であろうと思われる。即ち手指の姿勢をこれらの数値から推定し、指でつまんだ姿勢、もしくは手のひら全体でつかんだ姿勢のときに、手のひらの中心に配置したコライダーと物体が接触すれば、これを手のひらの中に固定する、という手法が採られているものと考えられる。

私は、指の間にコライダーを追加して「水かき」や野球のグローブのように手のひらを拡張して物体を包み込んでつかむことを考えた。またそれらのコライダーに物体を吸着させたり、物体とコライダーが近づくように物体に力を加えたりしてみた。そのいずれもうまくいかなかった。物体によけいな力や制約を加えることで物体はますます不安定に、挙動不審になってしまう。

Youtube などにアップされている Interaction Engine

のデモ映像を見ればおおよそその手法は推測できる。そこで私も、手指が物体に近接したときに物体を一時的に手のひらの子にし、手指が物体から離れたときに親子関係を解消する方法をとることにした。このように物体どうしを一時的に親子関係で拘束し、のちにその親子関係を解消するという手法は、ビデオゲームでは一般に用いられている（例えばキャラクターが移動する台の上に乗り降りするような場合）。Interaction Engine の原理はほぼ本稿の提案手法と同じだと考えられるが、その比較考察については後日十分に時間をかけて行いたいと考えている。

3. 手指による物体の把持、解放、投擲

本手法では、人差し指、中指、親指の先端にコライダーを装着し（「指先トリガー（finger-tip trigger）」と呼ぶ）、物体との当たり判定を行う。コライダーはそのままでは互いに反発して相手を弾き飛ばしてしまうので、指先トリガーは Trigger にしてある（Unity では Trigger は Collider の一種だが、Collider を Trigger にすると、当たり判定が発生しても物体どうしの反発力は生じない）。また手指につかまれる側の物体には本来のコライダーを包むように「外殻トリガー（outer trigger）」を用意して、つまりコライダーを二重にして、外殻トリガーを Trigger にしてある。言い替えると、本来のコライダー、つまり内殻コライダーは物体と同じ大きさであって物体の通常の当たり判定に用いる。外殻トリガーは物体よりも少し膨らませてあって、もっぱら指先トリガーとの当たり判定を行う（図 3、図 4 参照）。

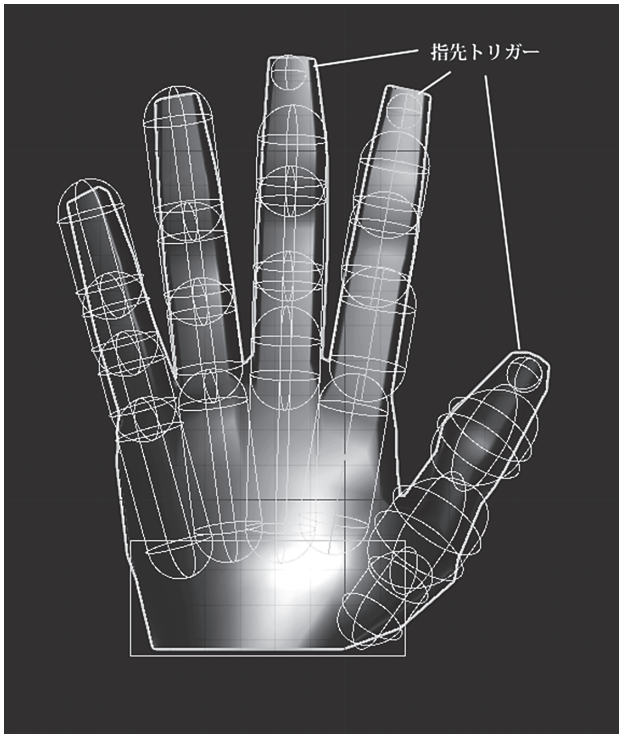

図 3　手指のコライダーと指先トリガー

図 3 に示すように、人差し指と中指と親指には、指先トリガー（finger-tip trigger）として、球形のトリガー（Sphere Collider）が取り付けてあり、それ以外のコライダーはカプセル形（Capsule Collider）になっている。指先トリガーの分、人差し指、中指、親指の指先のコライダーは短くなっており、逆に薬指と小指の指先のコライダーは長めにしてある。

片手の 3 つの指先トリガーがすべて外殻トリガーに接触したとき（OnTriggerEnter イベントが 3 つの指先で同時に発生したとき）、物体を手のひらの子にする。こうして物体は手の中に保持される（手のひらと連動して物体が動く）。

また、3 つの指先トリガーのうちの 2 つまでが外殻トリガーとの接触をやめたとき（OnTriggerExit イベントが 2 つの指先で発生したとき）、手のひらと物体の親子関係を解消する（物体が手のひらに連動しなくなる）。

ただ単に親子にしただけでは、コライダーどうしが干渉して物体が手のひらの中で暴れたり、弾き飛ばされたりする。またつかんだ後にも物体に運動量が残っていると物体がじりじりと動いたり、或いは重力の影響で落ちたりする。このために、物体を手のひらの子にすると同時に物体に対する物理エンジンを切る。すなわち物体の内殻コライダーを無効にして反発力が発生しないようにし、さらに物体に重力が働かないようにし（物体の Rigidbody の useGravity を false にする）、物体の速度と角速度を常に 0 に保つ（物体の Rigidbody の velocity と angularVelocity を 0 にする）。

指が開いて指先トリガーが外殻トリガーから離れ、親子関係が解消されれば、物体の内殻コライダーを有効に

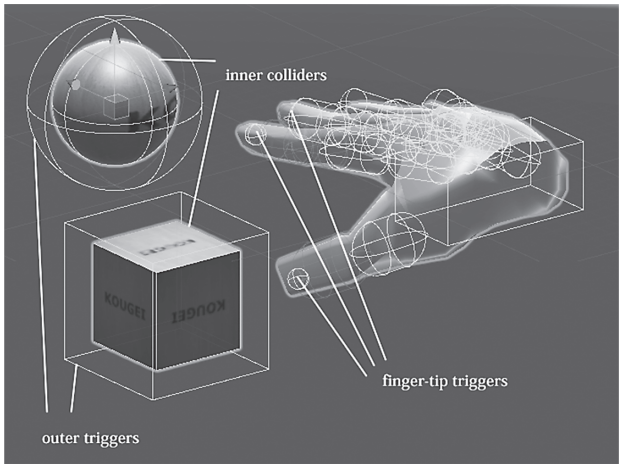

図 4　内殻コライダー（inner colliders）と外殻トリガー（outer triggers）

戻し、元通り物体に重力を働らかせ、物体には手のひらの速度と角速度を与える（つまり、手のひらを振った状態で物体を離すと物体を「投げる」ことができる）。

```
if (3 == countL && !isPinchedL && !isPinchedR)
{
      collider1.enabled = false;
      isPinchedL = true;
      Transform parent = L_Palm.transform;
      rb.useGravity = false;
      transform.parent = parent;
}
else if (2 > countL && isPinchedL)
{
      collider1.enabled = true;
      isPinchedL = false;
      transform.parent = null;
      rb.useGravity = true;
      rb.velocity = rblp.velocity;
      rb.angularVelocity = rblp.angularVelocity;
}
```

上記はC# で書かれたソースコードの一部であるが、countL は左手の指先トリガーのうち何個が物体と接触しているか、その個数である。isPinchedL は左手がなにか物体をすでにつまんでいるかどうかのフラグ（bool変数）。同様に isPinchedR は右手がつかんでいるかどうか。則ち、まだどちらの手もつかんでいなくて、左手の指先トリガーが3つとも物体に触れたときにまず isPinchedL を true にして、左手を物体の親にして、物体の内殻コライダーを無効にし、物体から重力の影響を無くしている。

またそれ以外のときで、左手がすでに物体をつかんでいて（isPinched == true）しかも左手の指先トリガーの接触数が2未満になった場合には、逆に、物体の内殻コライダーを有効にし、物体に重力を働かせ、手と物体の親子関係を解除する。同時に手の速度と角速度を物体に与える。

4. 実験結果

物体の内殻コライダーに対して外殻トリガーは1.4倍ほどの大きさ（相似比）にした。

手のひらから中指の先までの長さはおよそ20cm（Unity の記法で書けば0.2f。ここでf は浮動小数点数の意味）である。

たとえば1cm 角のサイコロや直径1cm（0.01f）のビー玉のようなものをLeap Motion を使って「両手のひらの上に盛る」ことは通常のコライダーの当たり判定だけで実現可能だが、その1粒を指先で「つまむ」ことはほとんど不可能である。1cm ほどの大きさの物体をつまむためには複数の指先をすべて1cm くらいの距離まで接近させなくてはならないが、Leap Motion の計測データにそれほどの正確さは期待できないし、指どうしを接近させると指先が互いにカメラの視界を遮り、オクルージョンが発生して指の位置を正確に認識できなくなってしまう。

しかしながら図5のように5cm（0.05f）角程度の立方体を「つかむ」ことはさほど困難ではない。Leap Motion の動作原理も知らない子供に体験させるような、インタラクティブアートのような非接触式の作品を制作するとき、操作が簡単で快適、かつ直感的であることには重要な意味がある。

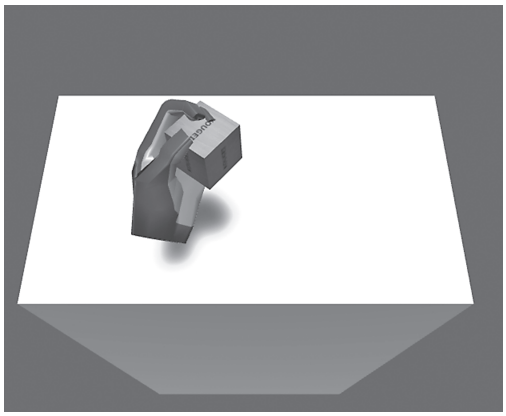

図5　立方体（5cm 角）

図6のように、手のひらにちょうど収まる程度の大きさの球体をつまむこともさほど難しくはない。

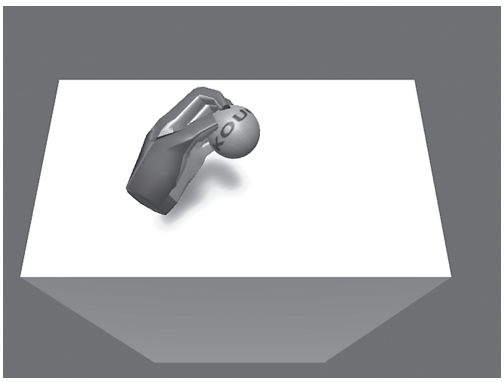

図6. 球（半径3cm）

図7のような平たい形状をつまむこともできる。ただしこの場合指が板を貫通してしまうことが多く、やや違和感がある。

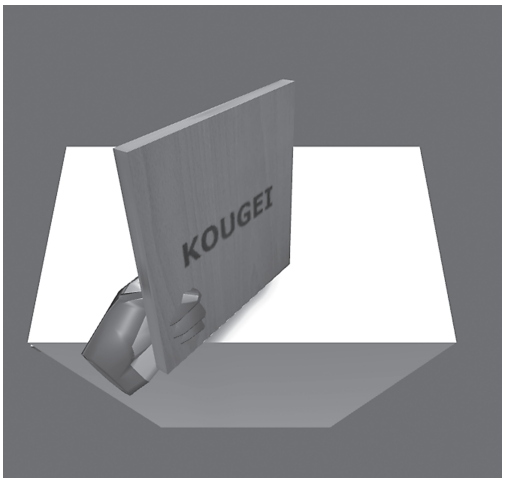

図 7　薄い板

図 8 のように細長い棒を持つこともできる。

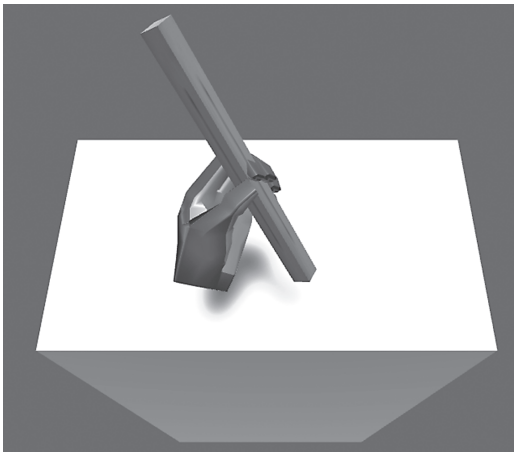

図 8　長い棒

本手法は、人差し指、中指、親指の先端による三点支持であるために、その三点がなす三角形が接触するような形状ならば、例えば図 9 のような手のひらよりずっと大きな物体でも、比較的安定して保持することができる。

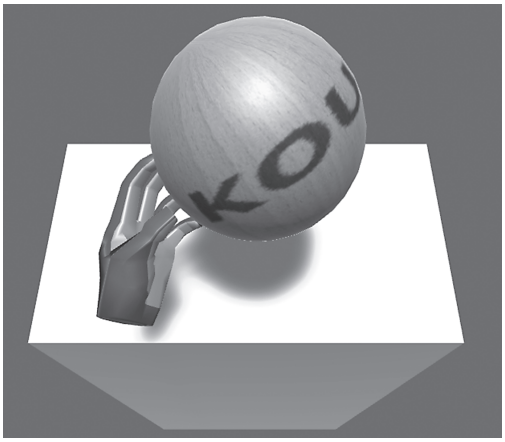

図 9　球（半径 10cm）

本手法では両手が同時に二つの物体を保持することは、親子関係の階層構造を壊すのでできないようになっている。そのため左手から右手へ物体を移す場合には、図 10 に示したように、まず左手でつかんで手のひらの上に物体を載せた状態で手を開き、右手でその物体をつかむことになる。

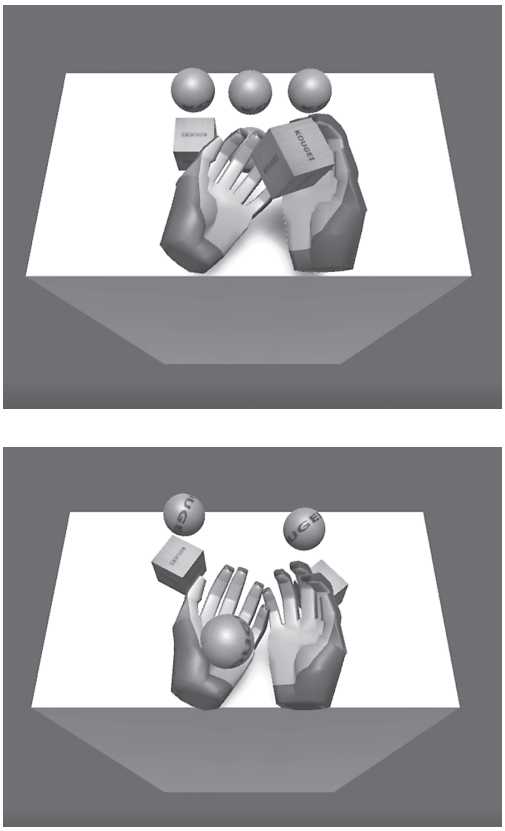

図 10　両手で物体を持ちかえる

以上の実験で物体の質量はいずれも大きさに関わらず 1kg にしてある。静止画像ではわかりにくいので動画 [17) 18)] を参照してもらいたい。

5.　考察

物体に内殻コライダーと外殻トリガーの二重のコライダーを仕込むという先行事例があるかどうか不明だが、おそらく Leap Motion Interaction Engine と本手法の最も異なる部分はこの二重コライダーであると考えられる。この二重コライダーをうまく使えば指が物体の中にめり込んだり貫通することのない、より自然なつまみ方ができるはずである。すなわち、指先トリガーが外殻トリガーに触れ、手のひらが物体を保持したときに、内殻コライダーを無効にするのではなく、内殻トリガーとして利用して、指が内殻トリガーと外殻トリガーの間に留まるように（つまり物体表面を覆う目に見えない薄皮に

指先が拘束されるように)、指の関節角度を制限すればよい。近い将来、このようなLeap Motionの関節角度を制御する手法を確立していきたいと考えている。

物を投げるとき、私たちは無意識に手首や指先のスナップを使っている。手のひらの速度を物体に与えるだけでは物はうまく飛んで行ってくれない。また、物体が手のひらから離れるタイミングがずれるとやはり物体はうまく飛んでいかない。指が物体から離れる（親子関係を解消する）というイベントと物体に速度を与えるイベントのタイミングを厳密に制御しなくてはならないが、そのためにはUnityのイベント処理の順序やタイミングに精通していなくてはならない。しかもこのことは物理エンジンやゲームエンジンごとにチューニングするする必要があり、同じゲームエンジンでもバージョンによる仕様変更もあり得ることに注意せねばならない。

Leap Motion開発者のブログによれば[19]、予備知識がないと、多くのユーザーは物体を持ったときに指を折り曲げて物体の中に深く突っ込んでしまう。そうするとその物体を放り投げるときにいつ物体が離れたか判定しにくいために、うまく操作できない、という（Without any affordances, users had a hard time identifying how to hold the objects. Once held, many users closed their hands into fists. This makes throwing more difficult as the object could become embedded into your hand's colliders after you release it.)。即ち現在のInteraction Engineでは指がめり込む問題を解決できていないことを意味している。

このブログではさらに、物体を放り投げたときに物体に速度を与えるためにPalmVelocityもしくはSlidingWindowの2種類のコントローラーを用意していると言っている（The PalmVelocity controller uses the velocity of the palm to decide how the object should move, to prevent the fingers from imparting strange and incorrect velocities to the object. The SlidingWindow controller takes the average velocity over a customizable window before the release occurred.)。PalmVelocityはその名の通りに、投げる物体に投げられた瞬間の手のひらの速度を与えるというものである。指の動きよりは手のひらの動きの方が突発的で不正確な動きを防ぐことができる。一方でSlidingWindowは手のひらの動きを0.1秒間平均した速度を物体に与えることで、手の動きのブレを防いでいるという。

このブログを読むとInteraction Engineの開発者たちも、やはり手の動きの不正確なブレや、物を投げるときに物が手を離れるタイミングなどに非常に苦労しており、指先のスナップなど細かな力加減を扱いかねていることがわかるのである。

オクルージョンによる誤判定などによって、手指姿勢の乱れが頻発するようなインタラクティブアートは快適に鑑賞し体験することができない。入力デバイスの計測ミスはある程度まで不可避であって、過度に信頼したり失望したりしても意味がない。誤判定とうまく折り合いをつけて誤判定に影響を受けにくい作品制作を心がけることも重要であると考えられる。

Interaction Engineのデモ映像などを見ると、仮想空間の中で指が器用に物体をつかみ、離し、投げられているように見えるが、それは開発者らがうまく操作するコツを知っているからであり、またうまくいった場合だけをつないで編集し、うまくいかなかった場合を見せていないからだと思われる。何の予備知識もなく初めて体験する人があれほど自然に物体を操作できるはずはないのではないか。このような技術をインタラクティブアートに応用するためには、老若男女、ありとあらゆるタイプの被験者がストレス無く仮装物体を手指で操作できるところまで､技術を洗練しなくてはならない。即ち､まだまだ多くの改良の余地が残されているということになる。

今後はUnityとUnrealで並行してLeap Motionの開発と作品制作を進めていき、最終的にはUnrealで学生たちとともに高品質なインタラクティブ作品を発表していきたいと考えている。

6. おわりに

インタラクティブアートにはさまざまな入力デバイスが用いられるが、それぞれに一長一短がある。Leap Motionは Unity や Unreal、vvvv[20] などさまざまなゲームエンジンやツールキットで手軽に利用可能であり、安価で壊れにくく信頼性も高く、電子工作の手間も要らない。計測精度が格別高いわけではなく、また画期的な仕組みが使われているわけでもないが、CG作品をメインで制作している学生がその技術を駆使してインタラクティブ性のある作品も作ってみる良いきっかけになると考えている。

すでに15年近く前になるが、私の研究室の卒業制作にKakomuという作品があり[21]、指導教員として指先の動きを3台のビデオカメラで撮影して検出するプログラムを手伝ったのだが、現在ならばLeap MotionとUnityを使えば簡単に実装することが可能である。こうした作品を作りたいという学生の要望はかつてもあったし現在もかなりある。今後もしばらくはLeap Motionや Unity、Unrealを用いてゼミの学生たちの期待に応えていきたい。

参考文献

1）Ian Failes: “Recalling Total Recall”, June 4, 2015
https://www.fxguide.com/featured/recalling-total-recall/#mocap

2）VF20th Anniversary Site: The Interviews
https://www.gamasutra.com/view/news/228512/Yu_Suzuki_recalls_using_military_tech_to_make_Virtua_Fighter_2.php

3）Takeo Kanade, Peter Rander, P.J. Narayanan: “Virtualized Reality: Constructing Virtual Worlds from Real Scenes”, Journal Article, IEEE Multimedia, Immersive Telepresence, Vol. 4, No. 1, pp. 34-47, 1997

4）井上英輝、永江孝規、長橋宏：“多視点動画像を用いた人体の姿勢のモデリング”、電子情報通信学会総合大会講演論文集 Vol. D-12、1998。

5）Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh: “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” In CVPR, 2017. arXiv:1611.08050 [cs.CV].

6）Perception Neuron by Noitom
https://neuronmocap.com/

7）永江 孝規、長橋 宏：“シルエット逆投影による多階調遮蔽包の計測”、映像情報メディア学会誌、vol. 52-4、pp. 616-622、1998。

8）Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, Christian Theobalt: “VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera,” ACM Transactions on Graphics（SIGGRAPH 2017）, Los Angeles, USA

9）James Steven Supancic, Gregory Rogez, Yi Yang, Jamie Shotton, Deva Ramanan: “Depth-based hand pose estimation: methods, data, and challenges”, International Journal of Computer Vision, Springer Verlag, 2018, 126（11）, pp.1180-1198.

10）Shanxin Yuan, Guillermo Garcia-Hernando, Bjorn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, Tae-Kyun Kim: “Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals”, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.

11）https://www.moguravr.com/leap-motion-2 デモ映像 Leap Motion Particle など。

12）Unity Assets for Leap Motion Orion Beta
https://developer.leapmotion.com/unity/#5436356

13）Leap Motion Unity Modules; Leap Motion’s Unity SDK 4.4.0
https://leapmotion.github.io/UnityModules/

14）Leap Motion Interaction Engine
https://developer.leapmotion.com/releases/interaction-engine-120

15）Leap Motion SDK for the Unreal Engine.
https://github.com/leapmotion/LeapUnreal

16）Leap Motion Unreal modules and example content.
https://github.com/leapmotion/LeapUnrealModules

17）Leap Motion + Unity 20181028 demo by NAGAE Takanori
https://www.youtube.com/watch?v=RUcSoWU2Xu8

18）Leap Motion + Unity demo 20181203 by NAGAE Takanori
https://www.youtube.com/watch?v=amnD1BBKycM

19）Weightless Remastered: Building with the Interaction Engine
http://blog.leapmotion.com/weightless-remastered-building-interaction-engine/

20）vvvv – a multipurpose toolkit
https://vvvv.org/

21）Gondaira Takuto, Nagae Takanori: “kakomu”
https://www.youtube.com/watch?v=tlCr9UriJho