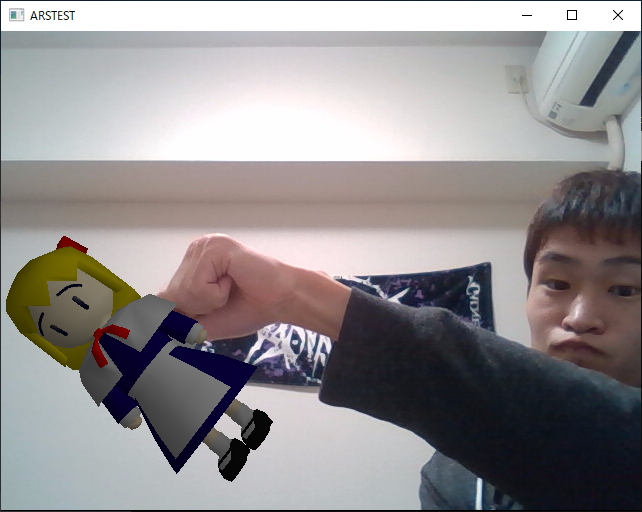
情報工学実験ⅠE2レポート

ゲームタイトル

ストレス発散！？人形サンドバッグ！！！！



5班

19x3132 根本颯汰

10月15日(木) 実施

1, プログラムの仕様

　今回のE2の実験では、映像とコンピュータグラフィックス(CG)の双方を組み合わせた複合現実(Mixed Reality)を用いたものをプログラムで作成した。

　具体的に、ユーザのライブ映像にCGである人形を映し出し、その停止した人形をユーザの指先で触れるとそれに応じてメトロノームのような動き(おきあがりこぼしの要領)をするようなプログラムを作成した。

2, 仕組み(状態遷移図)

　事前に撮影した背景画像と、現在の背景画像を別々に保存し、この二つの画像を画素ごとに比較して色の異なる部分を抽出する背景差分という手法を用いて背景とユーザの動きの区別を行った。

ユーザと人形(今回用いるオブジェクト)の接触判定は上記の通りに行うので、ユーザと人形の重なり部分を調べた。ノイズの除去を行うため重なり部分の重心と重なり部分が100px以上の時に接触しているとみなした。

　ユーザが人形に触れているかどうかをoverlappingでtrue/ falseを判定し、ユーザと人形が触れている状態のことをIN\_TOUCH、触れていない状態のことをOUT\_TOUCHとしたときの状態遷移図を以下の図１に示した。

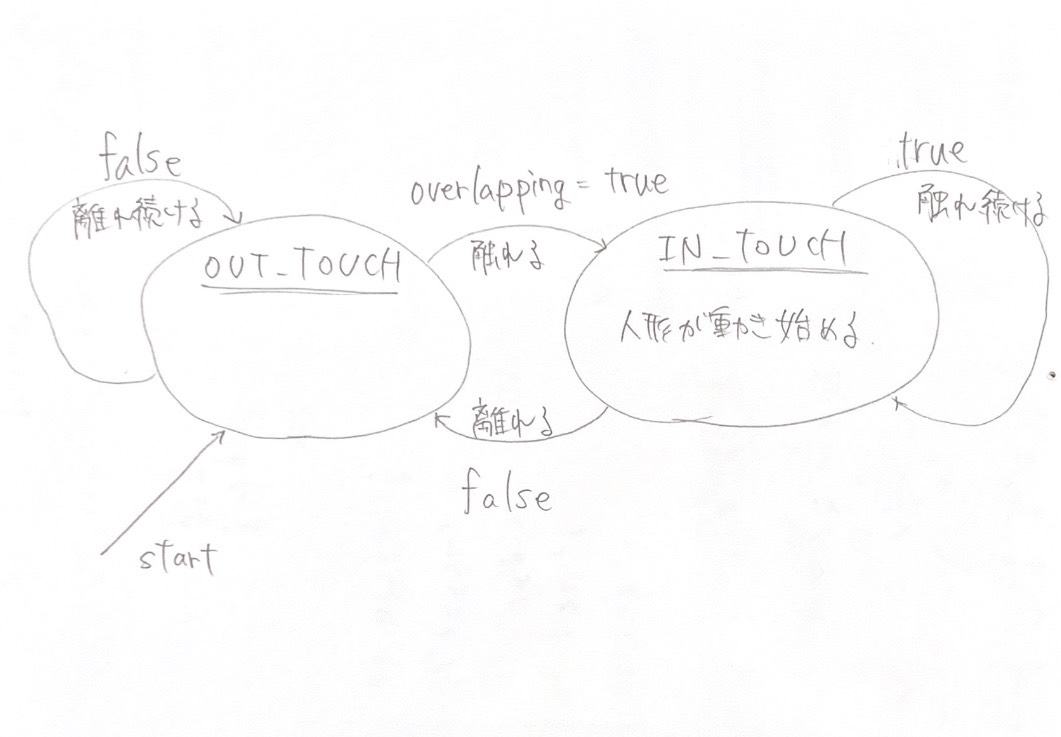


図１：人形とユーザの接触判定の状態遷移図

　初期状態をOUT\_TOUCHとしユーザと人形が重なったら(overlapping=ture)状態をIN\_TOUCHに移行しユーザと人形が離れたら、つまりユーザと人形が重なっていなかったら(overlapping=false)状態をOUT\_TOUCHに移行する。IN\_TOUCHのときにtrue、またOUT\_TOUCHのときにfalseの場合その状態を継続させた。

　また、人形の動作において不自然な動作にならないように人形が倒れている途中なのか、起き上がっている途中なのかを判断するgo\_back、人形の傾いている方向を判断するleft\_right、動いているときの時間をカウントする変数timeとしたときの状態遷移図を以下の図2に示した。(以下の状態遷移図において文字数が多く、見づらくなってしまうためgo\_backをG、left\_rughtをLと略記した)

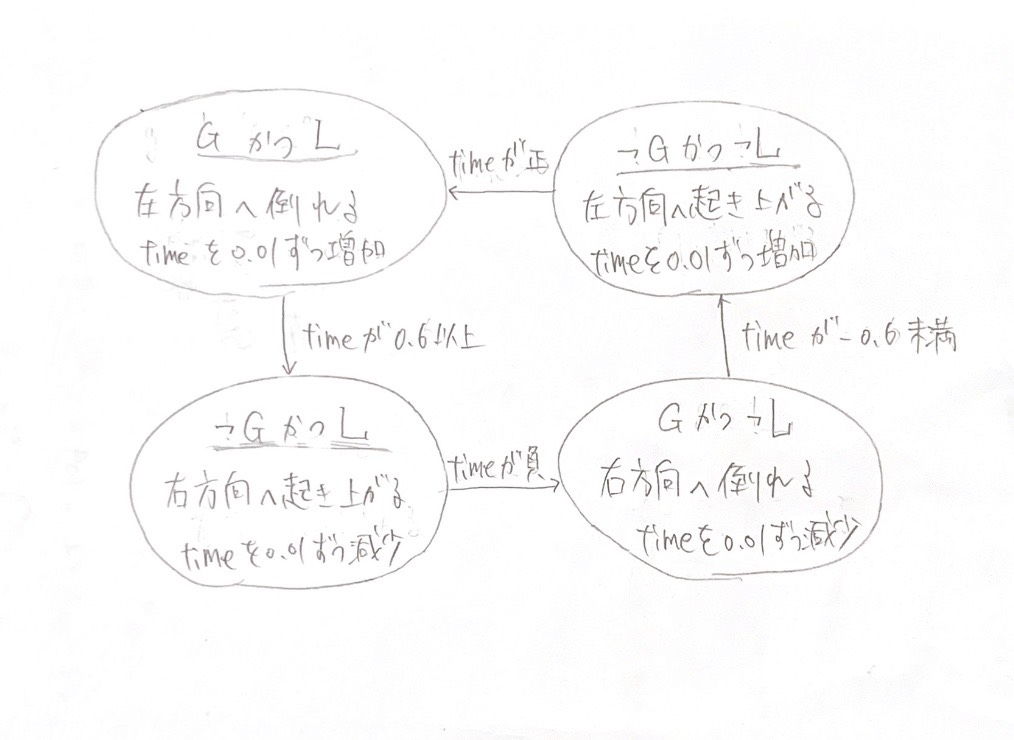


図2：人形の動きの状態遷移図

　図2の各状態においてtimeを増減させることで倒れ始め、起き始めからの時間を取得し、一定時間が経過したら次の状態に移行させる。timeを増加のみではなく減少もさせているのはtimeが0のときに垂直に立っている状態になっており、倒れきった状態の判定(図2の状態遷移図のtime>0.6, time<-0.6)がtrueになったときに0.6の値を0.075減少させることで徐々に倒れこむ角度が浅くなるようになっている。

　さらに、図2の各状態遷移において倒れる、起き上がるの動作は人形の中心にオブジェクトとしての座標があるため、人形の足元にオブジェクトの座標が来るように設定し、人形のサイズに合った半径を用いて半径×cosθと半径×sinθをx, y座標においてSetPositionしながらθの値を増減させることで人形の円運動を再現し、SetPositionのθの値に合わせてSetRotationすることで人形の足を中心の軸とする円運動を可能にした。この2つを駆使しておきあがりこぼしの運動を表現した。

3, 実現方法

このコードはVisual Studio 2017で行った。また、事前にdoll3.xを用意した。

Meshを継承したTouchableクラスを用意した。そのクラスの中に人形とユーザが重なりの部分とその重心を引数にし、接触判定をtrue/falseで返すcheckOverlappingと人形とユーザの重なり部分を引数に接触判定に応じた人形の動作を行うreactの２つのメンバ関数を用意する。他に直近の接触判定の結果を記憶するOUT\_TOUCH, IN\_TOUCHの列挙型とその列挙型の状態を保持する変数(state)をメンバ変数として用意した。

まず、背景差分を行う関数subtract\_maskを用意し、そこでは記憶した背景と現在カメラに写っている背景を引数にARSC::diffで双方の背景の差分を取得し、その結果をARSC::monochromeでモノクロ化(RGBを明度のみに変換)して、最後にその結果をthreshold関数で閾値と比較して閾値未満の場合0、閾値以上の場合画素値の最大値に変換するつまり白黒の二値化を行った。これにより背景を黒、ユーザの部分を白と区別することを可能にした。

次に、上記の通り定義したcheckOverlappingでは人形とユーザが重なりの部分とその重心を引数にし、ARSC::andで人形とユーザの重なり部分の中から閾値を超える部分のみで各画素ごとにand演算を行い接触部分を白くし、接触判定のtrue/falseを返す。また、その演算結果と先の背景差分から画素の位置の平均値と個数を取得し、重なり部分の重心をARSC::getCGで計算した。

さらに接触判定を行い、その結果に応じてメトロノームのような動きをさせる関数reactでは、図1の状態遷移図の通りIN\_TOUCHに状態が変化した瞬間の人形とユーザの重なったと判定し、動作の判定を担う変数stopをfalseにして人形が動き出すようにする。そのstopがfalseである限り人形はメトロノームのような動きをする。つまり、stopがfalseのとき図2の状態遷移図が動作する。その人形の動きは上の図2の通りである。

接触判定がtrueになり、stopがtrueになり人形が倒れ始める。そこからtimeが増加していき倒れる限界をあらかじめ設定してあるので(コード中では変数limit=0.6と置いている。図2では0.6のことを指す)その値を超えたら起き上がりの動作に移行するため、起き上がっている途中なのかを判断するgo\_backのtrue/falseを変更し、次に倒れこむときの角度を浅くするためにlimitの値を0.075減少させる。そして、起き上がりから倒れこみに変わるとき、つまりtimeの正負が変わるタイミングで起き上がっている途中なのかを判断するgo\_backのtrue/false と人形の傾いている方向を判断するleft\_rightのtrue/falseを変更する。これらのtrue/falseの変更とlimitの減少を繰り返すことで人形がおきあがりこぼしのような動作を行う。やがて、limitの値がとても小さくなり、timeの絶対値が小さくなったときに人形のおきあがりこぼしの動作をやめるためstopをtrueに変更して人形の動きをやめ、次回の動作の為にlimitを初期値の0.6に戻しておく。

倒れこみ、起き上がりの動作自体はmove関数で扱った。先の2,仕組み(状態遷移図)でも述べたようにx軸は半径×cosθ、y軸は半径×sinθずつ絶対座標でSetPositionさせ、SetPositionのθ値に合わせてSetRotationさせる。図2の状態遷移図に応じてθの増減を変換させるために-1, 1を掛けることでおきあがりこぼしのような人形の足元を軸にした回転を実現させた。

4, 実行結果

　ユーザが人形に触れるとそれに合わせて人形が倒れこみ始め、ある程度で起き上がりそれを繰り返し、倒れこみの角度が浅くなっていき、垂直になっているタイミングで止まるような動きになった。

5, 吟味

　今回のプログラムではおもちゃのおきあがりこぼしを複合現実上で再現しようと試みたものだった。実際のものとはおきあがりこぼしにつけるおもりによって起き上がり、倒れこみの速度が異なるが、それなりの再現度のものが出来たと思う。

しかし、今回作成したおきあがりこぼしはゆっくり起き上がるものであり、この速度を速めようとすると人形の足の位置が大幅にずれ、おきあがりこぼしではなく殴られて後方に飛んでいくような動作になってしまったため、倒れこみの速いものを作ろうと思うならばまだまだ改善の余地があると考えられる。

　また、起き上がりこぼしとは通常起き上がる際に垂直に立っているときに近いほど、起き上がる速度が徐々に速くなる。だが、今回自分で作成したものには起き上がりの途中に加速度を設けていない。加速度を加えて何度か実行してみたが、加速度の値が大きすぎると不自然になってしまい、逆に小さすぎると意味をなさず、体感なくても変わらないように思った。そこで、ちょうどいい値を模索したが見つからずまた、加速度を加えることでSetPositionに影響を及ぼし、これまた不自然な動作となってしまった。その結果として、今回は加速度を加えない方が最もおきあがりこぼしらしい挙動になることがわかったが、自分の模索方法や少し違ったアルゴリズム等を用いればより再現度が高まったかもしれない。この部分にも改善の余地がある可能性が考えられる。

　さらに、おきあがりこぼしの揺れる動作を減衰運動で表現すれば、先に説明した円運動を模倣した方法や条件分岐を用いずに、今よりさらに滑らかな挙動のものが作成できる可能性があると考えられる。