3O1-1in

仮想物理世界上で動く論理回路の実装

Logic gates in kinematic virtual environment

瀬戸口 幸寿　　成見 哲[*1]
Yukitoshi Setoguchi　Tetsu Narumi

[*1]電気通信大学 情報・通信工学専攻
Department of Communication Engineering and Informatics, The University of Electro-Communications

Each individual piece of artificial life (ALife) is usually controlled by the AI, which is programmed by the scientist. Therefore, we couldn't say the piece lives completely in the virtual environment. We propose ALife which works within the kinematic virtual environment including the program itself. As a first step of it, logical units in the kinematic environment are implemented using Unity, a game constructing tool with kinematic engine. A ring oscillator works faster than the previous one which was made by PhysX, a lower-level kinematic engine. A comparison of the speed of a simple particle-collision calculation between Unity and CUDA, a lowest-level engine with graphics processing unit, showed that CUDA is much faster than Unity for simple calculation. Also a simpler NAND gate is proposed to build complicated logical units easier.

1. はじめに

既存の人工生命 (Artificial Life, ALife) の研究では, 人工生命の個体がある仮想空間内で駆動するのに対し, 個体の運動や思考を支配する AI はプログラマによって設計されたプログラムによって制御されている. その為既存の人工生命の挙動には, 設計者によって定められた境界条件が存在し, その枠を超えた挙動を示すことが少ない. そこで別の方針として,AI を人工生命のモデルと同様の空間で定義することが考えられる. リアルな環境を構築し, その微妙な差異による影響を受けることによって,AI が挙動に再現性や規則性を持たず, より生命的な人工生命が生まれる可能性がある. 本研究では, 人工生命の AI の基本的な部品となる論理回路を Unity[1] による仮想物理世界上に実装した. 仮想物理世界とは重力や弾性力が働くコンピュータ内の世界である.

William[2] は空間中のグリッドに固定して用いる数種類の立方体に, 信号の伝送や論理作用素, 駆動パーツといった役割を設けて論理回路や動きを作りだし, 自己複製する機械の研究を行った. 本研究では一種類の立方体で論理設計と駆動の両方を目指す. また空間中の格子点に限ることなく,3 次元空間を自由に動けるようにして, 仮想物理環境の複雑な影響を受け取ることにより, その挙動に複雑性が生まれることを期待する. なお, 自己複製は本研究では考えないものとする.

原らは PhysX[3] を用いて仮想物理環境を構築し, 機械式論理回路を実装した [4]. さらに研究内でより大きな回路の駆動を確認すると共に, 壊れやすく動きが遅いことから回路の安定化と高速化の必要性を提唱した. 本研究では,Unity を用いることで機械式論理回路の構造面から安定化, 高速化に成功した. また簡易な物理演算を行うプログラムを実装し, 剛体シミュレーションにかかる計算時間を Unity と比較することで, 数値計算の観点から高速化ができる可能性を示した.

連絡先: 成見 哲, 電気通信大学　情報・通信工学専攻, 〒182-8585 東京都調布市調布ヶ丘 1-5-1, Tel:042-443-5340, Fax:042-443-5384, Mail:narumi@cs.uec.ac.jp

2. Unity

Unity[1] は米国 Unity Technologies が提供するゲームエンジンである. ゲーム制作用の統合開発環境であるが, 物理計算やレンダリングを自動で行い, 仮想物理世界上に存在するモデルを GUI で設定できる為, 本研究では機械式論理回路の実装に用いている.

一方で,Unity は物理エンジンに PhysX を用いているが, PhysX は剛体シミュレーションに対してハードウェアアクセラレートに対応しておらず, 物理計算の最適化には対応していない. 今後, より大規模な論理回路や仮想物理世界上の人工生命を駆動させる為に, 物理計算の最適化が必要であると考えられる.

3. 機械式論理回路の実装

AI を仮想物理世界上で実装する為に,Unity による仮想物理世界上で機械式論理回路, とりわけ NAND ゲートを実装する.

3.1 箱型 NAND ゲートの実装

原らによる研究 [4] では, 論理回路が壊れる, 入力から応答まで時間がかかるといった問題があった. そこで, 論理回路の可動部を大きく制限し, 箱で保護することで耐故障性と高速化を目指した. 図 1 はその方針のもとに作られた箱型 NAND ゲートである. 箱型 NAND ゲートは半透明の紫色の箱によって, 論理回路同士の干渉により壊れないように保護すると同時に, 可動部の遊びを限定することで高速な駆動を実現している. 下部に位置する赤と青は入力ロッドと呼び, 外部から押されている状態が入力 1, そうでない状態が入力 0 である. 緑色の剛体は出力ロッドと呼び, 飛び出ている状態を出力 1, そうでない状態が出力 0 である. この箱型 NAND 回路を用いて, 原の研究

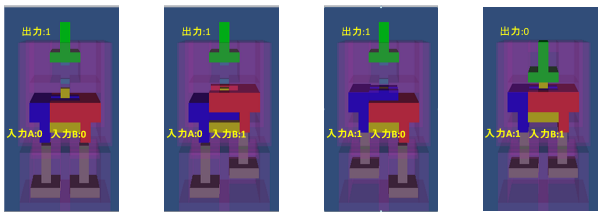

図 1: 箱型 NAND ゲート

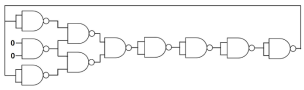

図 2: リングオシレータの回路図

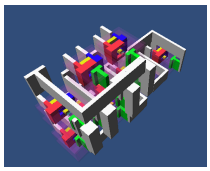

図 3: 実装したリングオシレータ

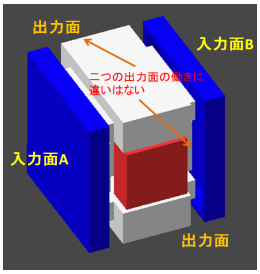

図 4: 立方体型 NAND ゲート

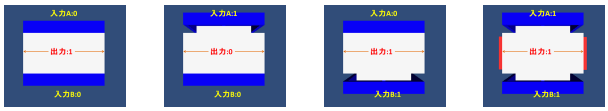

図 5: 立方体型 NAND ゲートの動作例

で NAND 回路の速度を測定するのに用いられたものと同様のリングオシレータ回路 (図 2) を実装し, 速度の比較を行った. 図 3 が実装した回路の様子であり, その周期はおよそ 5 秒である. 原の研究における等価な論理回路の周期が約 128 秒である為 [4], 大幅な高速化に成功したと言える.

3.2 立方体型 NAND ゲートの実装

箱型 NAND 回路は既存の機械式 NAND 回路に対し, 安定かつ高速に駆動することが確認できた. その一方で,

- NAND ゲート同士を繋ぐ配線パーツが一意に定まらない
- ゲートが空間に固定されているので人工生命の駆動部品となることが期待できない

といった欠点が存在する. 目的となる人工生命の AI を構成する為には, 対故障性, 駆動速度をそのままに, 論理作用素としてだけでなく配線パーツや駆動部品としても働くことが望ましい. また, 空間に固定されるのではなく, 自由に動き回っても意味を失わない回路であることが, 生物の固体を模すためには不可欠である.

以上の条件を満たす論理回路を図 4 に示す. これを立方体型 NAND ゲートと呼ぶ. 青の面を入力面と呼び, 赤の面を出力面と呼ぶ. 図 4 の裏面にも出力面があり, 二つの出力面の働きに違いはない. これらは, 箱型 NAND 回路における入力ロッド, 出力ロッドに対応する. ただし, そのまま配線パーツとして用いることを考慮して, 入力面は押し込まれたら 0, そうでなければ 1 とし, 出力面は出っ張ったら 0, そうでなければ 1 とする. 図 5 に示す入力例は左から,(入力 A, 入力 B, 出力)=(0,0,1),(1,0,1),(0,1,1),(1,1,0) を表す.

4. 剛体シミュレーションの高速化

回路を大規模化すると,Unity の動作が遅くなるのを確認した. これは剛体シミュレーションにかかる時間が大きくなった為と考えられ, 剛体シミュレーションの高速化は課題の一つである. 本研究では, 剛体同士の衝突判定を行う方法として剛体を微小な球体の集合で近似する手法 [5] に着目し, 球体同士の

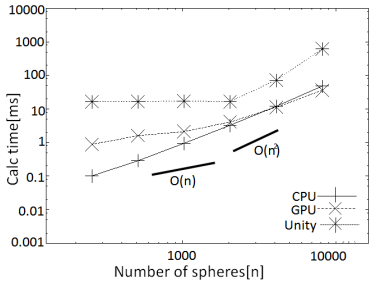

図 6: 速度比較

剛体シミュレーションに対し最適化を行った簡易な物理計算プログラムを作成し, 同様の仮想環境下で Unity によるものと 1 ステップあたりにかかる計算時間の比較を行った. 作成した物理計算プログラムでは, あらかじめ粒子の座標をソートすることによって衝突判定の負担を減らし CPU で計算を行う CPU モードと, 全ての粒子に対して GPU を用いて並列に衝突判定を行う GPU モードを設け, それぞれ計算時間の測定を行った. GPU でのプログラムには CUDA[6] を用いた. Unity,CPU モード,GPU モードに対し, 粒子数ごとの 1 ステップに費やす計算時間のグラフを図 6 に示す.Unity は CPU モードに比べ 10 倍程遅く, 球の数が増えると GPU モードが最も速くなった.

5. まとめと今後の課題

Unity を用いて機械式論理回路の高速化, 安定化を構造面から実現した. また, より生物らしい人工生命の AI を駆動させることを目標として, 論理作用素としてだけでなく配線パーツ, 駆動部品としても働くような機械式論理回路の提案を行った.

また, 独自に剛体シミュレーションコードを作り比較したところ, Unity より高速に計算できることが確認された. つまり, 機械式論理回路の為の物理計算の最適化を行うこと, また GPU による並列計算を行うことにより, 回路の高速化, 大規模化が図れる可能性を示した. しかし, 本研究で作成したプログラムは球体同士の衝突に限られ, 論理回路の駆動は実現できていない. また, 立方体型 NAND ゲートを組み合わせて自由に動き回る個体を再現する予定である.

参考文献

[1] Unity Technologies Japan『Unity Game Engine』http://japan.unity3d.com

[2] A self-replicating programmable constructor in a kinematic simulation environment, William M Stevens (2011)

[3] PhysX http://developer.nvidia.com/object/physx.html

[4] 原健一郎, 成見哲, 小林聡『高性能計算を用いた仮想物理世界での論理回路の実現』(電気通信大学 2013 情報・通信工学専攻修士論文)

[5] 原田隆宏, 田中正幸, 越塚誠一, 河口 洋一郎『GPU を用いたリアルタイム剛体シミュレーション』(情報処理学会研究報告 2007)

[6] CUDA ZONE by NVIDIA. https://developer.nvidia.com/what-cuda