人工生命体の発生と運動を同一の枠組みで実現する動力学的グラフモデル

Wriggraph: A Kinetic Graph Model That Uniformly Describes Ontogeny and Motility of Artificial Creatures in a Single Framework

電気通信大学電気通信学研究科
人間コミュニケーション学専攻
佐山研究室
0436011 佐野 浩司

0436011 Kouji Sano
Sayama Laboratory, Department of Human Communication
University of Electro-Communications

sano@cx.hc.uec.ac.jp

概要

従来の動力学的人工生命モデルは，形態形成と運動の制御を別々の枠組み内で表現している．特に，運動の制御機構を形態に埋め込んでいないため，物理実装の現実性を低くしている．その問題を解決するため，人工生命モデル Wriggraph を提案する．Wriggraph は，ノードとエッジからなるグラフ型動力学的グラフモデルである．ノード同士の局所的な相互作用のみを用いて，生命体の発生と運動の制御を同一の枠組み内で記述する．Wriggraph は他の類似する人工生命体モデルと比較し，構造的な現実性が高く，物理実装の可能性が高い．本稿では，Wriggraph を用いて人工生命 “テトラ” を構築した．テトラは，一定のノード数を持つように形態発生し，構成要素のノードを人為的に取り除くと，元のノード数まで再生できる．また，運動機関を機能分化させ，シミュレーション空間内に設定された座標に向かって，進行方向を自律的に調整しながら移動できる．これらの能力は，すべて構成要素間の局所的な相互作用のみを用いて達成される．今後の課題として，進化的計算などを取り入れて，構成要素としてより多数のノードを持ち，より複雑な動作が可能な人工生命体を構築すること，また，モデルの改良により，ノードの複製についてより現実性を高めることがあげられた．

Abstract

Earlier models of kinetic artificial creatures typically describe ontogenetic mechanisms and motion control mechanisms separately. The latter are often assumed free from morphological constraints, reducing the biological and physical plausibility of these creatures. To go over these limitations, here we propose “Wriggraph”, a kinetic graph model that describes both ontogeny and motility of graph-based artificial creatures in a unified framework. The utility of this model is demonstrated by constructing an artificial creature *tetrad*, which can spontaneously develop into a stable tetrahedral shape, move using differentiated nodes, home toward a predefined goal, and self-repair in events of node removal. All of these capabilities are achieved collectively by functionally identical nodes and their local interaction only. We expect a number of future extensions of this research project, including the automated acquisition of more complex morphologies and behaviors using evolutionary computation, and the revision of model assumptions to make node division mechanisms more realitsic.

目 次

図 目 次

表 目 次

第1章　はじめに

自然界の生命体は，非常に完成されたシステムといえる．例えば形態発生の段階においては，たった一つの細胞から成長がはじまり，細胞を増やしながら体構造を複雑に形成，変化させ，ついには一体の生物を完成させる．また，形態発生とともに運動器官や神経系，感覚器官を備え，それらを連動させることで，食料を探し回り，危険からの脱出を行うことが出来る．さらに，体組織が破壊されても，ある程度なら再生を行い，元の姿に戻ることが出来る．このように生命体は驚くほど精密な体組織を形成，維持し，周囲の環境に柔軟に対応しながら活動できる．

生命体の基本単位である細胞の内部および細胞間では，化学反応を主体とした局所的な相互作用が起きている．それらの要素間の相互作用が上記のような生命活動の源であり，生命体は中央集権的な制御装置がなくとも，個体全体を維持することが出来る．

そのため，複雑な組織形成や，環境に対する柔軟さなどといった生命体の持つ特性を，人工物にも取り入れようとする研究は近年盛んになっている．そして，生命システムの制御機構である局所的な相互作用を人工システムの制御に利用しているものが出現している．例として，Skype[23] などといったインターネット上における P2P システムがあげられる．P2P システムでは多数のネットワークノードが相互接続している．ノード同士が局所的に情報をやり取りすることで，トラフィックを考慮した効率のよいルーティングやノードの検索などを実現している．ただし，P2P システムにおける相互作用はノード間の情報通信といった論理的なものに限られている．一方，実世界においてシステムの要素またはシステム全体の物理的な構築や運動などについて制御する実例は少ない．

しかしながら，局所的な相互作用を利用してシステムを物理的に制御する人工物は今後必要になるだろう．例として，極地や宇宙開発 [28] の現場に要求されるロボットが挙げられる．これらの現場では，人の手によるコントロールを必要とせず，あらゆる状況に自動的に対処する必要があり，また不慮の事故に対する堅牢性を持ちつつ環境に対して柔軟に対応できることが必要である．これらのような現場で必要となる装置に求められる性質は，局所的相互作用を利用している自然界の生命体がもつ性質と同等であるといえる．

そのため，局所的相互作用によりシステムを物理的に制御する人工生命モデルやロボットモデルの研究が多く存在する．既存の人工生命モデルの多くは，体構造はシミュレーション空間に表現しているが，運動の制御機構は空間内に表現していない．このため，物理的実現性や構造的現実性が低いといえる．また，体構造を発生させる形態発生の段階と，形態発生が完全に終了した後に活動を開始する運動の段階が明確に区別されており，人為的に二つの段階を切り替える必要がある．また，構成要素の故障に対する耐性について言及している人工生命モデルは存在しない．これらと同様なことがロボットシステムの研究にもいえる．

そこで本研究では，以上のような既存のモデルの問題点を克服し，局所的相互作用のみ

で物理的な形態発生や運動についてシステム全体を制御し，環境に対して柔軟に対応できるような人工生命を記述できるモデル Wriggraph を提案する．

本稿の構成を次に述べる．第 2 章で関連研究について述べる．第 3 章では関連研究の問題点を克服するような新しい動力学的人工生命モデル Wriggraph を提案する．第 4 章では Wriggraph を実装して実験を行うためのコンピューターシミュレーションについて述べる．第 5 章では Wriggraph の有用性を確認するために，シミュレーション空間内において設定された課題を解決できる人工生命体をデザインしていく過程を紹介する．第 6 章ではデザインした人工生命体を通じて，本稿の結論と今後の課題を述べる．

第2章　関連研究

動力学的要素を考慮した 3 次元空間内における人工生命シミュレーションは，Sims による Box Creature[20, 21, 22]，Komosinski らによる FramSticks[9, 10, 11, 12]，O'Kelly らによる BubbleGene[17, 18]，Hornby らによる L-System Creature[5, 6]，Bongard らによる AO Creature[2, 3, 4] が挙げられる．

Box Creature[20, 21, 22] は図 2.1 のように直方体が木構造的に連結した外観を持つ．生命体は遺伝型を解読し表現型となる形態を発生させる．遺伝型は木構造に類似した記述方法を用いており，発生する生命体の形態も木構造である．生命体は，立方体同士の連結部が曲がったり，回転することで運動する．この運動は形態の遺伝型とは別の遺伝型より表現されるニューラルネットにより制御されている．

BubbleGene[17, 18] は図 2.2 のように球が木構造的に連結した外観を持ち，Box Creature と同様な発生方法をとり，同様に形態と別の遺伝型により表現されるニューラルネットで運動を制御する．

FramSticks[9, 10, 11, 12] は図 2.3 のように円柱が木構造的に連結した外観を持つ．Box Creature と同様な発生方法をとる．また，運動のための機関が表現されている．運動は形態と同じ遺伝型により表現されるニューラルネットで制御されている．

L-System Creature[5, 6] は図 2.4 のような外観を持ち，L-System[13] を遺伝型の記述に用いている．運動は形態と同じ遺伝型により表現されるニューラルネットで制御されている．

AO Creature[2, 3, 4] は，図 2.5 のような球体ユニットが接続している外観を持つ．球体は各種のセンサーやニューロン，接続部を回転させるモーターを備えている．それらの機能発現や抑制を制御するのは遺伝子産物である．遺伝子産物は特定の遺伝子が発現することにより放出される仮想的な化学物質である．遺伝子産物は，球体ユニットの複製やさまざまな表現型の発現，そして他の遺伝子の発現を制御する．また，遺伝子産物がユニット間を拡散することにより，ユニット内の他の遺伝子が発現して別の遺伝子産物が生成され，さらにそれに適合する表現型を発現したり，別の遺伝子が活性化されるという遺伝子調整ネットワークを用いている．

先行するこれらのモデルには，以下のような問題点がある．

(1) いずれのモデルも形態が木構造状に発生する．そのため，木構造以外の形態は表現できない．

(2) 発生過程における構成部品の欠損などのエラーについて考慮しているモデルはない．

(3) 形態発生と運動が別々の段階で表現されている．そのため，形態発生の段階と運動を行う段階は人為的に切り替える必要がる．また，AO Creature は構成要素の複製を人為的に停止しなければならず，形態発生の制御が人工生命体の内部システム内で完結していない．

(4) ニューラルネットにより人工生命体の運動を制御しているモデルでは，ニューラルネットを局所的な制御機構ではなく，単一の脳として用いられている．このためこれらのモデルは局所的な分散システムではなく，中央集権的なシステムといえる．

(5) ニューラルネットは論理的に存在しており，物理的に構成要素の中に埋め込まれておらず，シミュレーション空間に表現されていない．また，[2, 3, 4] ではモーターやセンサー，ニューロンなどのユニットの運動制御部は，ユニット内の固定された座標点に表現されるようになっている．しかし，同一座標点に複数の制御部品が表現できるようになっているため，構造の現実性が低い．

特に (5) について，人工生命の形態についてはシミュレーション空間内に表現しているが，その形態の運動を表現するモーターなどの稼動部や，運動を制御する神経組織などの要素を空間内に表現していない人工生命モデルは，embeddedness[25] の度合いが低い．embeddedness とは，生命モデルなどについて用いる指標である．活動空間において，モデルの要素についての状態情報がどの程度表現されているかを示す [19]．embeddedness が低ければ，モデルの構造的な現実性や，物理実装の可能性が低い．

次に，実世界において相互作用によりシステムを物理的に制御する例として，GOLEM[14, 15] や M-TRAN[7, 8, 16] を挙げる．

GOLEM[14, 15] は図 2.6 のような外観を持つ．モデルの形状はコンピューターシミュレーションを通じて作成される．Box Creature 等のモデルのように，形態は遺伝型より表現し，運動の制御には，体構造の外部に用意されるニューラルネットを利用している．モデル形状がシミュレーションにより作成された後，rapid prototyping machine を用いてモデルを物理的に実装している．

M-TRAN[7, 8, 16] は図 2.7 のように多数のモジュールでロボットを形成している．ニューラルネットを内包するモジュール間の相互作用により，与えられた特定の形状について移動運動パターンを生成することができる．

これらのシステムの問題点は以下のとおりである．

(1)GOLEM は，運動を制御するニューラルネットが物理的な体構造に埋め込まれておらず，外部に計算機が必要である．そのため個体として不完全である．

(2) どちらのシステムも自己組み立て等の形態発生については運動とは別の行動制御プログラムが必要であり，二つの段階を人為的に切り替える必要があるため，自律性が低い．

(3) 発生過程における構成部品の欠損などのエラーについて考慮されていない．

以上のように，形態発生と運動の双方を同一の枠組み内に表現し，自然界の生命と同様に構成要素間の局所的な情報通信や力学的相互作用により自律的に発生や運動を決定することに完全に成功した例は無く，構成要素のエラーに関する考慮を示す例も無い．そこで本研究では，上記を達成することを目指して，embeddedness が高く，物理的実世界における応用を視野に入れた人工生命体モデル Wriggraph を提案する．

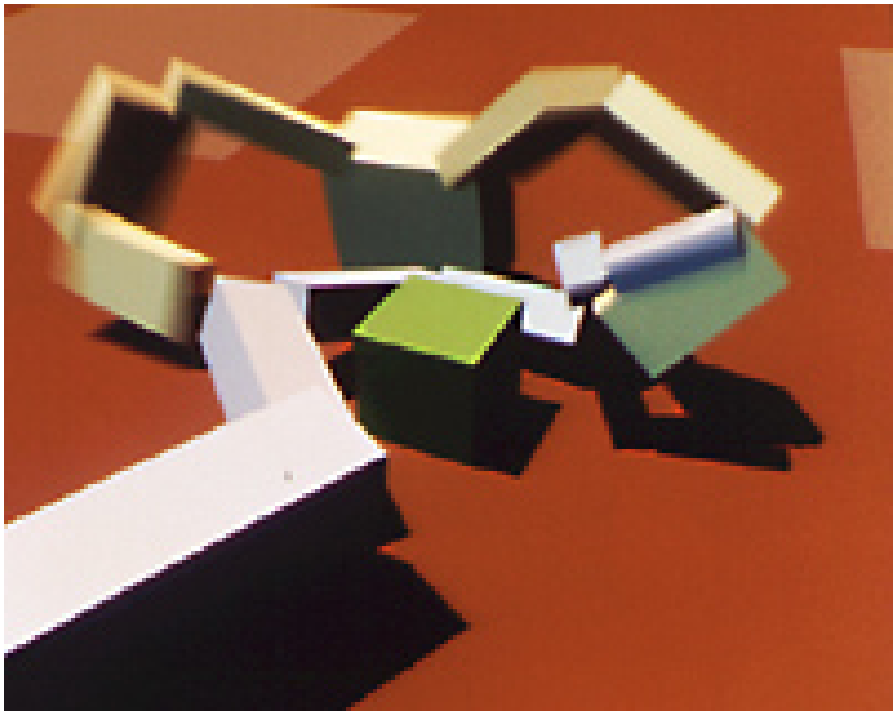


図 2.1: Box Creature[20, 21, 22] における人工生命体の例．([22] より引用)

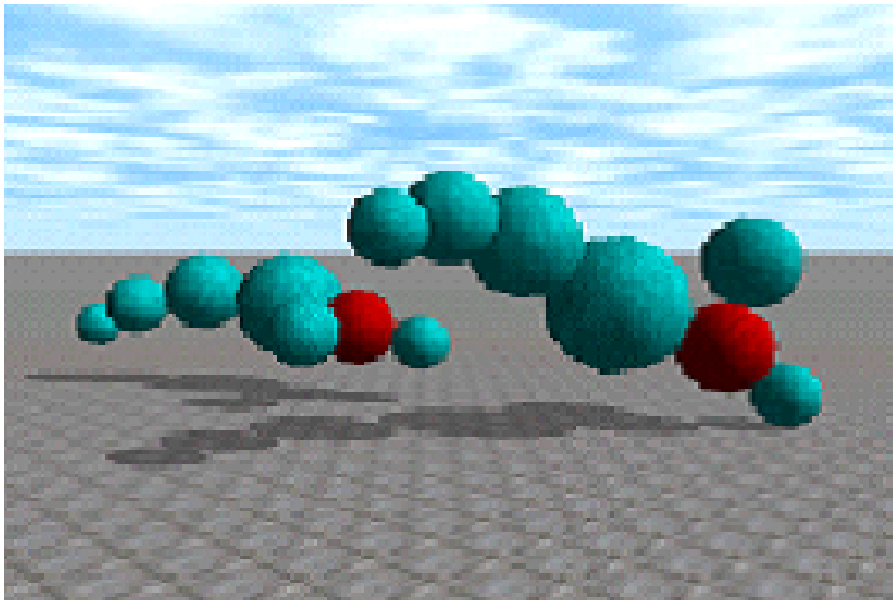


図 2.2: BubbleGene[17, 18] における人工生命体の例．([18] より引用)

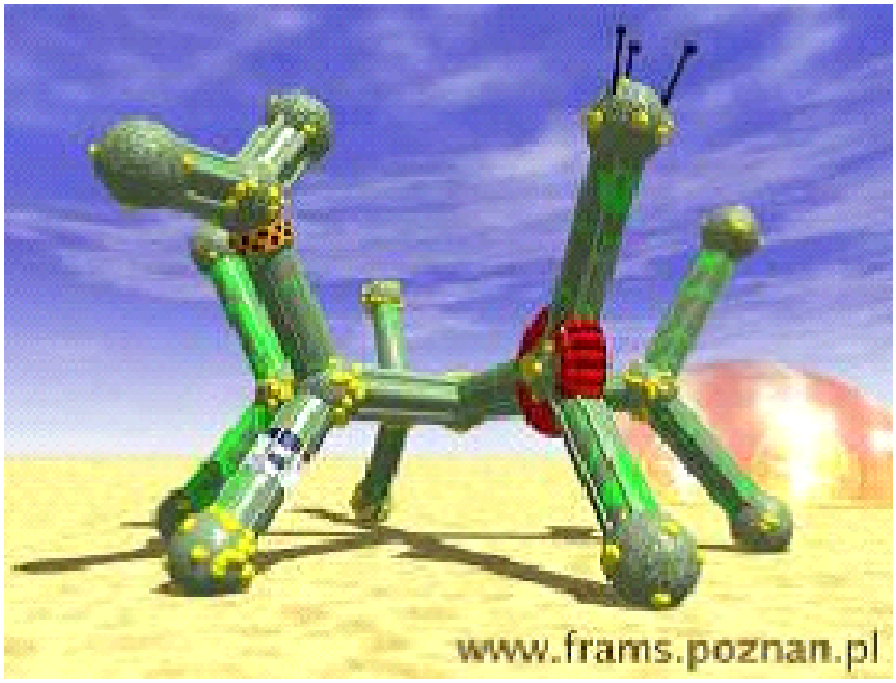


図 2.3: FramSticks[9, 10, 11, 12] における人工生命体の例．([12] より引用)

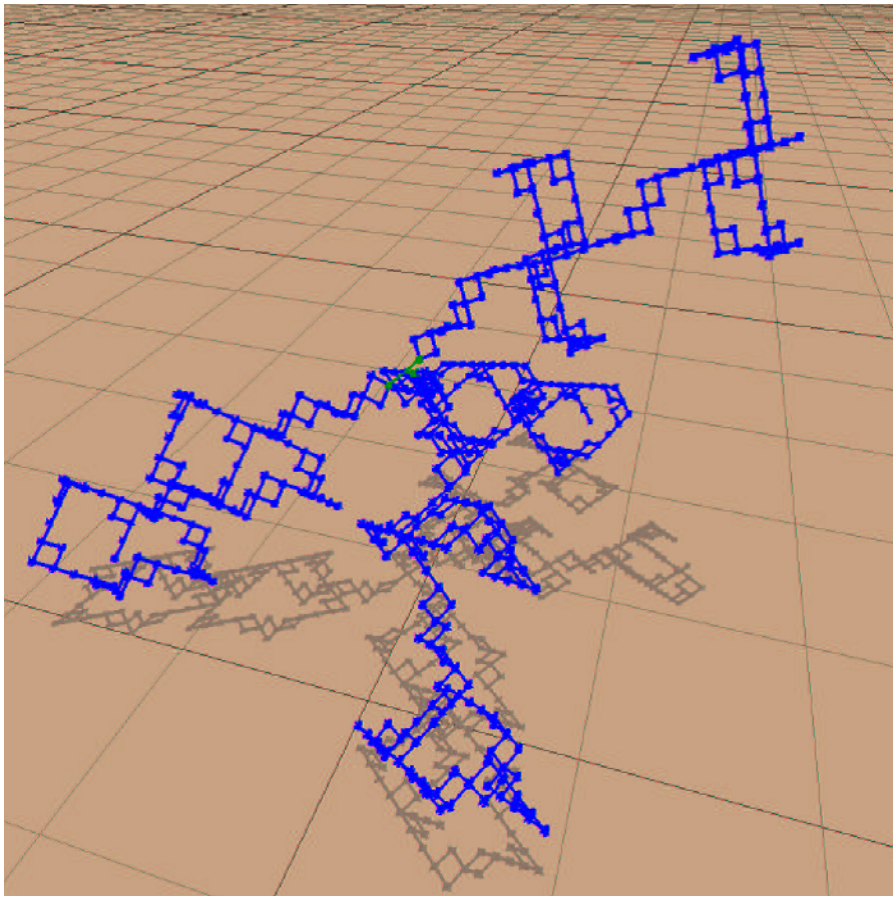


図 2.4: L-System Creature[5, 6] における人工生命体の例.

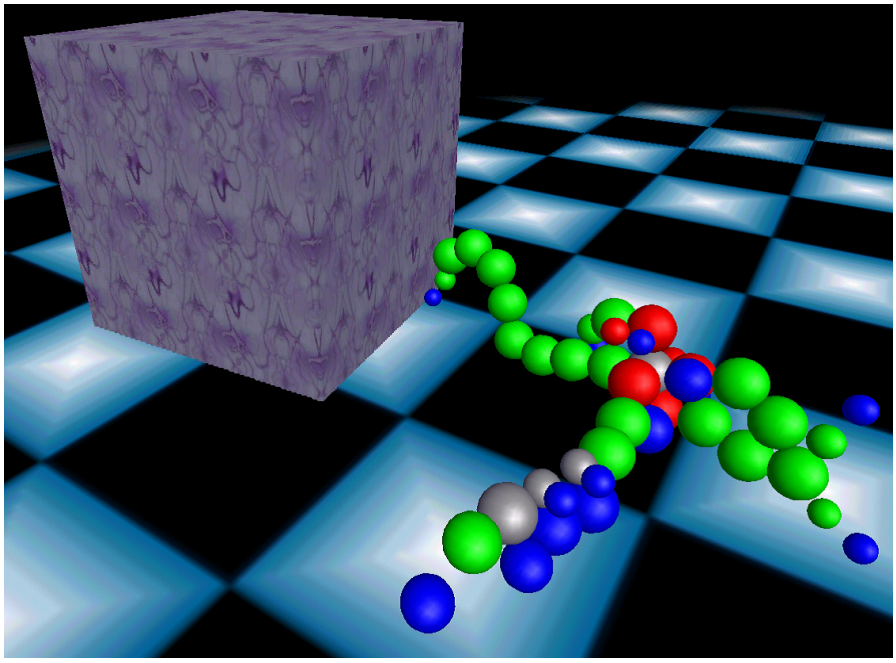


図 2.5: AO Creature[2, 3, 4] における人工生命体の例．([4] より引用)

図 2.6: GOLEM[14, 15] におけるモデルの例．([15] より引用)

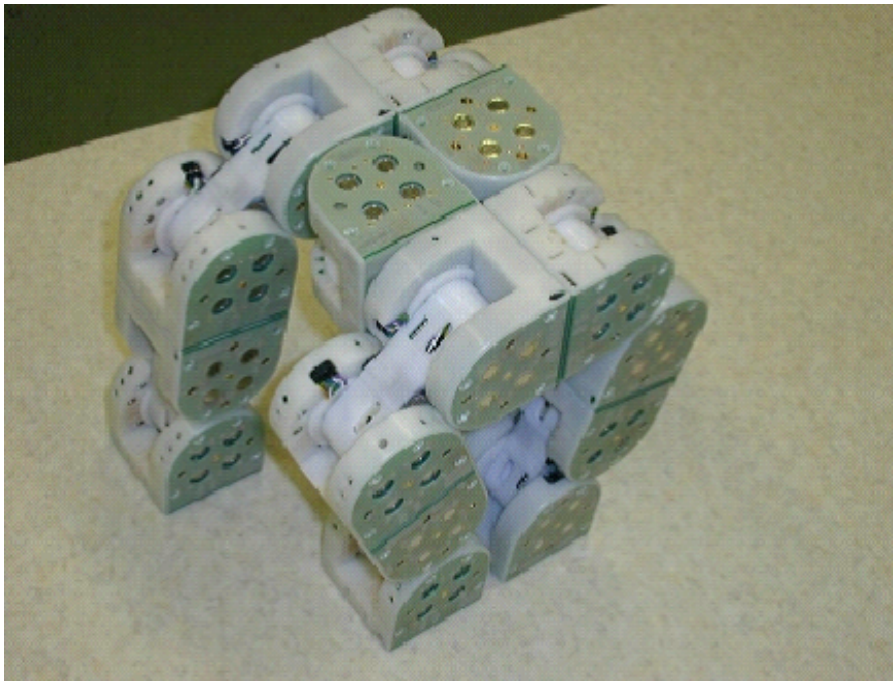


図 2.7: M-Tran[7, 8, 16]. ([16] より引用)

第3章　Wriggraphの詳細

本章では，本研究で提案する動力学的人工生命体モデル Wriggraph の詳細について述べる．Wriggraph は，既存の人工生命モデルなどに比べて embeddedness が高く，また形態形成と運動の双方を同一の枠組み内に記述し，自然界の生命と同様に構成要素間の局所的な情報通信や力学的相互作用により発生や運動を制御できるモデルを目指す．

3.1　Wriggraphの概要

Wriggraph は Sodaplay[24] やグラフオートマトン [27] などのようなグラフ型モデルである．図 3.1 のように，人工生命体の形態は，三次元空間内にノードとエッジにより自由に構成される．

個々のノードは自然界の細胞を模しており，共通の遺伝子コードと，N 種類の仮想的な化学物質を持つ．化学物質は，ノードに接続するエッジを通じて隣接するノードに拡散する．また遺伝子コードは，ノード内の化学物質がどのように化学反応するか，そして化学物質の量の変化によりどのようにノードを複製して形態発生を行うか，どのようにエッジを伸縮させるか等を指示する．

このことから，Wriggraph は極度に分散化したシステムであるといえる．つまり，遺伝子コードを調整することにより，個々のノード内における化学物質の量が安定したり，化学物質の量が振動するような系を記述することが可能である．それらの内部状態の振る舞いを通じて，ノードやエッジの物理的な状態を変化させ，人工生命体のマクロな挙動を決定する．

3.2　Wriggraphの動力学的特性

ノードは，動力学的パラメーターとして，位置，速度，加速度，一定の質量，静止摩擦係数，動摩擦係数，ノードに接続するエッジの標準の長さを持つ．また，エッジはばねの性質を持ち，パラメーターとして，ばね定数，標準の長さを持つ．

エッジの長さの標準値と実際の長さの差を Δx とし，k をばね定数すると，エッジに接続しているそれぞれのノードには

$$F_{edge} = k\Delta x$$

の力が掛かる．この力によりノードの位置を変化させる．なお，二つのノード間に張られているエッジの標準的な長さは，それぞれのノードにおいて遺伝子により設定されたエッジの長さの相加平均となる．

また，ノード同士の距離を保ち，人工生命体の形を維持するために，エッジで接続されていないノード同士には斥力を掛ける．斥力係数を r とし，ノード間の距離を l とすると，それぞれのノードには

$$F_{repulsive} = r\frac{1.0}{l}$$

のような斥力が掛かる．

また，ノードには速度に比例する抵抗が掛かる．速度を V，速度抵抗係数を a とすると，速度抵抗は以下のようにして求まる．

$$F_{airresist} = -aV$$

3.3 ノードの内部情報

ノードは内部情報として，隣接ノードリスト L，N 種類の化学物質に関する情報，遺伝子コードといった三種類の情報を持つ．(図 3.1)

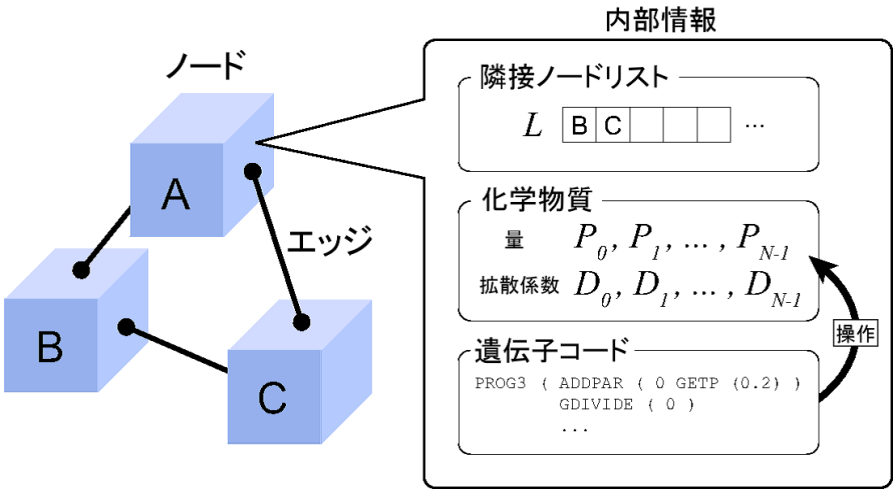


図 3.1: Wriggraph による人工生命体の概観とノードの内部情報．図ではノード A とノード B とノード C が相互接続している．ここではノード A の内部情報 (隣接ノードリスト，遺伝子コード，化学物質の量 P と拡散係数 D) が表示されている．接続ノードリストにはノード A に接続しているノード B とノード C への参照が格納されている．

隣接ノードリスト L には，そのノードにエッジで接続しているノードへの参照が格納される．参照の並び順は，ノードにその隣接ノードが接続された順番である．

N 種類の化学物質に関する情報は，化学物質の量 P_i と拡散係数 D_i を保持する ($i = 0, 1, ...N-1$)．P に格納される数値は 0 から ∞ の実数である．D に格納される数値は 0 から 1 までの実数である．

隣接するノード間では，エッジを通じて化学物質の拡散が起きる．拡散の割合は，それぞれのノードにおける D によって決まる．あるノード A における各化学物質の単位時間当たりの変化量は，次の式によって算出する．

$$\Delta P_i^A = \alpha \sum_{x \in L_A} \sqrt{D_i^A D_i^x}(P_i^x - P_i^A)$$

L_A は，ノード A の隣接ノードリストをあらわす．ここでは，双方のノードの D_i の相乗平均を用いている．このため，一方のノードの D_i が 0 であれば，拡散は生じない．また，α は ΔP が大きすぎる数値になるのを防ぐ係数である．本稿で示すシミュレーションでは $\alpha = 0.01$ を用いた．

遺伝子コードは人工生命を構成するノードが共通に持つ遺伝子であり，本研究で独自に作成した模擬関数型言語のプログラムコードである．遺伝子コードは P や D の値を参照，操作する．そして，それらの値をノードやノードに接続しているエッジの物理的パラメーターに写像したり，その値がある閾値を越えればノードの複製等のノードの挙動を引き起こすというような条件を設定する．遺伝子コードの例をリスト 3.1 に示す．遺伝子コード内における定数は，0 から 1 の実数値である．遺伝子コードで利用可能な関数の一覧を表 3.1，3.2，3.3，3.4，3.5，3.6，3.7，3.8 に示す．

リスト 3.1: 遺伝子コードの例

```
PROG3 (
        TDIVIDE (
                ADDPAR (
                        0
                        0.9
                )
        )
        ADDPAR (
                0.2
                MUL (
                        0.5
                        GETP (
                                0
                        )
                )
        )
        PROG2 (
                SUBPAR (
                        0
                        MUL (
                                GETP (
                                        0.2
                                )
                                0.7
                        )
                )
                ADDDIF (
                        0.2
                        0.008
```

30)
31)
32)

表 3.1: 算術関数.

関数名	引数	説明
NEG	(x)	$-x$ を返す.
ABS	(x)	$\|x\|$ を返す.
POW2	(x)	x^2 を返す.
POW3	(x)	x^3 を返す.
POW4	(x)	x^4 を返す.
ADD	(xy)	$x+y$ を返す.
SUB	(xy)	$x-y$ を返す.
MUL	(xy)	xy を返す.
DIV	(xy)	x/y を返す．y が 0 なら 0 を返し，DIE 関数を実行する.
MIN	(xy)	x と y のうち小さいほうを返す.
MAX	(xy)	x と y のうち大きいほうを返す.

表 3.2: 論理演算関数.

関数名	引数	説明
NOT	(x)	x が 0 なら 1 を返す．x が 0 以外なら 0 を返す.
AND	(xy)	x と y が 0 でなければ y を返す．どちらか一方もしくは両方 0 の場合は 0 を返す.
OR	(xy)	x と y のうち 0 でないほうを返す．両方 0 でないときは x を返す．両方 0 なら 0 を返す.
LT	(xy)	x より y が小さければ 1 を返す．そうでなければ 0 を返す.
GT	(xy)	x より y が大きければ 1 を返す．そうでなければ 0 を返す.

3.3.1 遺伝子コードによる P，D の操作

関数には，P や D の個々の値 P_i，D_i を操作したり，参照するものがある．そのため，それらの関数では与えられた引数を，添え字 i を指定する情報として用いる．このとき，関数の引数はあらゆる実数値をとりうるが，化学物質の種類が N 種類の場合，添え字の範囲は 0〜$N-1$ でなければならない．よって，添え字 i を以下の関数 I を用いて求める，

$$i = I(v) = \lfloor Ntanh(v) \rfloor$$

このとき，$i < 0$ の場合は $i = 0$ とする.

表 3.3: 実行制御関数.

関数名	引数	説明
IFT	(xy)	もし x が 0 でなければ y を実行し，y の値を返す．x が 0 であれば 0 を返す.
IFTE	(xyz)	もし x が 0 でなければ y を実行し，y の値を返す．x が 0 であれば z を実行し，z の値を返す.
PROG2	(xy)	x，y と逐次に実行し，y の値を返す.
PROG3	(xyz)	x，y，z と逐次に実行し，z の値を返す.

表 3.4: 化学物質操作関数.

関数名	引数	説明
GETP	(x)	$P_{I(x)}$ を返す.
GETD	(x)	$D_{I(x)}$ を返す.
SETP	(xy)	$P_{I(x)}$ に y を代入する．また，x を返す.
SETD	(xy)	$D_{I(x)}$ に y を設定する．また，x を返す.
ADDPAR	(xy)	$P_{I(x)}$ に $d(y)$ を足す．また，x を返す.
SUBPAR	(xy)	$P_{I(x)}$ から $d(y)$ を引く．また，x を返す.
ADDDIF	(xy)	$D_{I(x)}$ に $d(y)$ を足す．また，x を返す.
SUBDIF	(xy)	$D_{I(x)}$ から $d(y)$ を引く．また，x を返す.

ADDPAR，ADDDIF，SUBPAR，SUBDIF は P や D の値に引数の値を加算，引算する関数である．この関数によって P や D の値が急激に変化することを避けるため，加算，引算する引数の値を v とすると，次に示す関数

$$d(v) = \beta tanh(v)$$

で得られる値を加算，引算している．本稿で示すシミュレーションでは $\beta = 0.01$ を用いた．また，演算した結果，P，D の値の範囲を超える場合には，値を切り捨てて範囲に収まるようにする．IMPULSE 関数でも同様にして他のノードの P に値を加算している．PULSE 関数の場合は値を直接加算している．

表 3.5: 発信関数.

関数名	引数	説明
PULSE	(x)	$P_{I(x)}$ が 1 以上で あれば，$P_{I(x)}$ をそのノードの隣接ノード数 $\times 2$ で割り，結果の値を隣接ノードの $P_{I(x)}$ に加算する．そして，自分の $P_{I(x)}$ を 0 に設定する．また，$P_{I(x)}$ の値に関わらず x を返す.
IMPULSE	(xy)	$P_{I(x)}$ が 1 以上であれば，そのノードの隣接ノードの $P_{I(x)}$ に $d(y)$ を加算し，自分の $P_{I(x)}$ を 0 に設定する．また，$P_{I(x)}$ の値に 関わらず x を返す.

表 3.6: 形態形成関数.

関数名	引数	説明
GDIVIDE	(x)	$P_{I(x)}$ が 1 以上であればノードを複製し，$P_{I(x)}$ を 0 にする．また，$P_{I(x)}$ の値に関わらず x を返す.
TDIVIDE	(x)	$P_{I(x)}$ が 1 以上であればノードを複製し，$P_{I(x)}$ を 0 にする．また，$P_{I(x)}$ の値に関わらず x を返す.
DIE	-	遺伝子コードの実行後，自ノードを削除する．0 を返す.
CUTEDGE	-	ノードに接続しているエッジをすべて切断する．切断するエッジがなければ 0 を返す．エッジを切断できれば 1 を返す.

表 3.7: 環境情報関数.

関数名	引数	説明
SMELL	-	シミュレーション空間内に設置された目標座標への距離の逆数を返す.
GROUND	-	ノードが地面に接していれば 1 を返す．そうでなければ 0 を返す.

3.3.2 人工生命体の形態発生

人工生命体の形態発生は，既存のノードから新しいノードが複製されて，エッジで相互接続することによって起こる．この操作は，GDIVIDE 関数と TDIVIDE 関数により実行される．ノードが複製されると，複製された子ノードと，複製元の親ノードとの間にエッジが張られる．このとき，親ノードに接続している他のノードがあれば，そのノードと子ノードとの間にもエッジが張られる．これは，生命体に 3 次元的な構造を持たせるため必要な操作である．GDIVIDE 関数と TDIVIDE 関数では，エッジの張り方に違いがある．(図 3.2, 図 3.3)

GDIVIDE 関数と TDIVIDE 関数において，親ノードがほかのノードと接続していない場合，子ノードは親ノードの近くのランダムな位置に設置される．逆に，エッジを張れるノードがある場合，親ノードと，親ノード以外に最初にエッジを張るノードの中間の座標にベクトル b を足した位置に子ノードを設置する．中間座標はエッジが通っているため，b によ

表 3.8: 動力学的パラメーター操作関数.

関数名	引数	説明
ELENGTH	(x)	$l = (l_{max} - l_{min})tanh(P_{I(x)}) + l_{min}$ をエッジの長さとする．また，x を返す.
DYNFRIC	(x)	$\mu^d = (\mu^d_{max} - \mu^d_{min})tanh(P_{I(x)}) + \mu^d_{min}$ をノードの動摩擦係数とする．また，x を返す.
DIFFRIC	(x)	$\mu^s = \mu^d + (\mu^f_{max} - \mu^f_{min})tanh(P_{I(x)}) + \mu^f_{min}$ を静止摩擦係数とする．また，x を返す．このとき，$\mu^f_{max} = \mu^s_{max} - \mu^d_{max}, \mu^f_{min} = \mu^s_{min} - \mu^d_{min}$ である.

り子ノードの位置をずらす必要がある．本稿のシミュレーションでは，b の各座標には-1 から 1 までの乱数を用いている．図 3.2，図 3.3 では，見易さのために子ノードの位置はその原則を無視している．

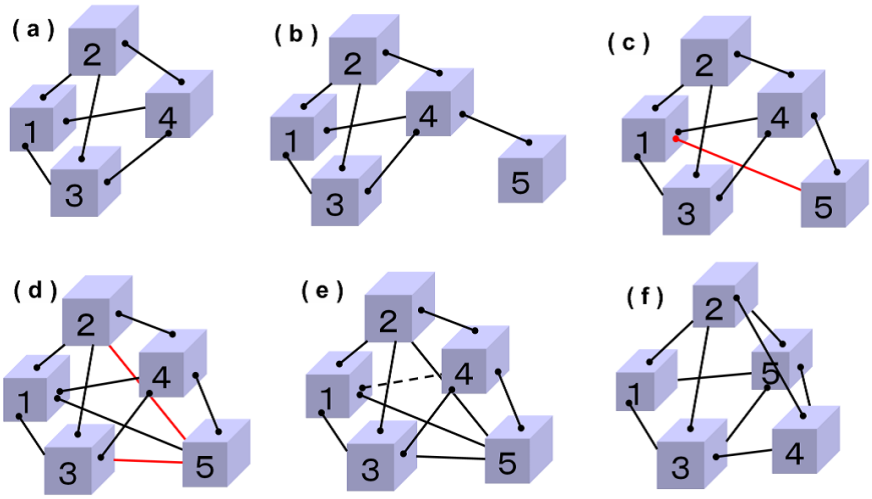


図 3.2: GDIVIDE によるエッジ接続の様子．図 (a) は，新しくノードが複製される前の様子である．ノードに記された番号は，そのノードが生成された順番を示している．ノード 1 は一番古いノードであり，ノード 4 が一番新しいノードである．それぞれのノードはエッジで接続されている．ここでは，ノード 4 が新しくノードを複製する．このとき，以下の 4 段階の操作が行われる．(1) 親ノード（ノード 4）が子ノード（ノード 5）を生成し，子ノードと親ノードとの間にエッジを張る (図 (b))．(2) 親ノードにエッジを張っているノード（ノード 1，ノード 2，ノード 3）のうち，一番次数の大きいノードと子ノードとにエッジを張る．新しく張るエッジを赤色で示す (図 (c))．この場合ではどのノードの次数も同じなので，任意のノード（ノード 1）が選ばれている．(3) 親ノードにエッジを張っているノードのうち，次数の少ない順に最大 2 個のノード（ノード 2，ノード 3）を選び，子ノードとエッジを張る．新しく張るエッジを赤色で示す (図 (d))．(4)(3) でエッジを張ったノードの数が 2 個ならば，子ノードとエッジを張っている最大次数のノード（ノード 1）と親ノードとのエッジを切断する．切断するエッジを点線で示す (図 (e))．図 (f) が最終状態である．図 (a) において相互接続していたノード 1，ノード 2，ノード 4 の間に新しいノード 5 が割り込むように位置しており，新しいノードは既存のノード接続を押し広げるような形になることがわかる．この複製方法では，ノードは生命体の表面積を広げるように増加する．

ノードが複製された際には，親ノードから子ノードへ要素の遺伝が生じる．遺伝子コードは直接コピーされる．化学物質については，D はコピーされるが P はコピーされず，子ノードの P は 0 で初期化される．このことにより，ノード同士の間で内部状態に不均質性が生じ，機能分化を生じさせる．

3.3.3 人工生命体の運動

人工生命体の運動は，エッジが伸縮したり，ノードと地面との間に摩擦力が掛かることにより生じる．これらは，関数 `ELENGTH`，`DYNFRIC`，`DIFFRIC` によりノードやエッジの物理的パラメーターが変化することにより生じる．これらの関数において，物理パラメーターには引数 v から変換した添え字 i で得られる P_i を適用するが，P_i の値の範囲は 0〜∞ であるので，物理的パラメーターに適用するのは現実的に不適当である．そこで，各パ

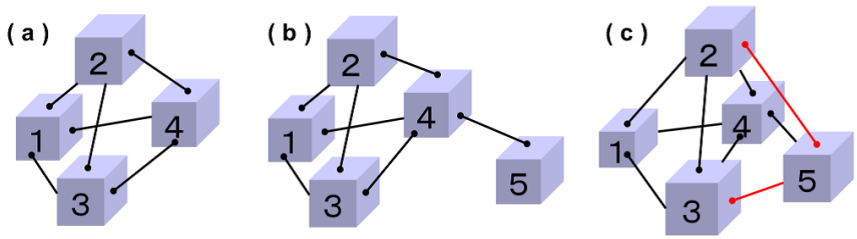


図 3.3: TDIVIDE によるエッジ接続の様子．GDIVIDE と同様に，図 (a) を新しくノードが複製される前の状態であるとする．このとき，以下の 2 段階の操作が行われる．(1) 子ノードと親ノードとの間にエッジを張る (図 (b))．(2) 親ノードの接続ノードリストの最初の 2 個 (2 個なければ最初の 1 個) のノードを選んで子ノードと接続する．この場合，ノード 4 の接続ノードリストの最初の 2 個はノード 3 とノード 2 である．新しく張るエッジを赤色で示す (図 (c))．この方法では，生命体に四面体が連結するようにノードが増加する．

ラメーターの最小値と最大値をあらかじめ設定し，以下のようにして得られる値 x を実際のパラメーターとして設定する．

$$x = (\text{最大値} - \text{最小値})tanh(P_i) + \text{最小値}$$

遺伝子コード内で，物理的パラメーターを変化させる関数が呼ばれない場合は，設定されている最小値を物理的パラメータとして設定する．

3.4 Wriggraph の特徴

Wriggraph は人工生命体の活動に必要な要素はすべて明示的に表現している．人工生命体を形作る形態はノードとエッジである．運動機関は伸縮するエッジである．人工生命全体の運動を制御するのは，エッジによる値の拡散や，発信関数などによる局所的なノード間の情報伝達である．これらはすべて単一の枠組みの中に埋め込まれており，相互に作用して変化できる．よって，Wriggraph の embeddedness の度合いは高く，物理的実現性や構造的な現実性が高いといえる．

第4章　Wriggraphのシミュレーション実装

本章では，Wriggraph を実装して実験により有用性を確認するためのシミュレーションについて述べる．

4.1　シミュレーターの実装

シミュレーターの実装には C++言語を用いる．物理シミュレーションは，単位離散時間ごとに運動方程式を近似的に数値積分するオイラー法を用いる．3 次元 CG の表現には DirectX[1] を利用する．遺伝子コードの実装には，線形 GP[1, 26, 32] を参考にしている．シミュレーターは，Windows2000 以降と DirectX9.0 以上で作動する．筆者の Web サイト[2]で無料で入手可能である．シミュレーターと同じディレクトリに「source.txt」という名前で遺伝子コードのファイルを設置し，シミュレーターを起動すると，ファイルから遺伝子コードを読み込み，空間内に一つのノードを表示し，そのノードに読み込んだ遺伝子コードを持たせる．その他の操作法を表 4.1 に示す．

4.2　シミュレーション空間の詳細

人工生命を構成するシミュレーション空間は，境界のある直方体形をした有限の 3 次元空間であり，左手座標系を用いる．境界面では物体は非弾性衝突を行う．空間には-Y 方向に一定の重力がかかる．シミュレーション空間を図 4.1 に示す．図の下部に一つのノードが配置されている．本シミュレーションでは，ノードを立方体として表示した．Wriggraph において，ノードは力学的に質点として扱うので，ノードの形状は任意である．また，空間の奥にひし形の物が置かれている．これは，第 5 章でデザインする人工生命体の移動目標座標の目印となる．そのほかに，空間内には空と地面が表示される．シミュレーターがあるディレクトリ内の bmp ファイルを変更すると，これら空間内に表示するジオメトリのテクスチャーを変更できる．

シミュレーションに用いた各種のパラメーターを表 4.2 に示す．

4.3　シミュレーションの流れ

シミュレーションは，以下の流れを単位時間ごとに行う．

[1] http://www.microsoft.com/japan/windows/directx/default.mspx
[2] http://sayama1.hc.uec.ac.jp/~sano/work/wriggraph/

表 4.1: シミュレーターの操作法.

操作	動作
マウスの左クリックドラッグ	視点の回転
左 Shift ＋左クリックドラッグ	視点の平行移動
C キー	全ノードの位置の平均座標に注視点を合わせる
右 Shift ＋左クリックドラッグ	注視点を中心に視点を回転
C キー＋右 Shift ＋右クリックドラッグ	ノードの平均座標を常に向きながら視点を回転
C キー＋左クリックドラッグ	ノードの平均座標を常に向きながら視点を回転
スペースキー	すべてのノードを削除し，遺伝子コードファイルを読み直して新しい一つのノードを設置
F キー	視線方向にカメラを前進
B キー	視線方向にカメラを後退
H キー	ノードの動きや遺伝子コード実行を停止．再度 H キーを押すと再開.
S キー＋ノードを左クリック	ノードの P と D の現在の量を表示．量が 0 のところは表示しない．ノードがない場所で S キー＋左クリックすると表示を消す.
K キー＋ノードを左クリック	ノードを削除
カーソルキー上	1 フレーム表示あたりの，遺伝子コード実行と物理シミュレーション，P の拡散の回数を上げる
カーソルキー下	1 フレーム表示あたりの，遺伝子コード実行と物理シミュレーション，P の拡散の回数を下げる

1. ノード間のすべてのエッジごとに，化学物質の拡散の移動量を決定する．また，ノードの現在位置からとエッジの標準の長さから，ばねの法則によりノードにかかる力を計算する.

2. すべてのノードごとに，遺伝子コードを実行する．遺伝子コードにおいてノードが複製された際，新しいノードとそれにエッジで接続するノードとの間の拡散の移動量を決定する．自ノード以外のノードの関数 (IMPULSE 関数や PULSE 関数) による P の変化を記録しておく.

3. すべてのノードごとに，速度抵抗やノード間の斥力を計算し，1 で計算された力や地面との反発，摩擦力とともにノードの速度，位置に適用する.

4. すべてのノードごとに，すでに決定されている拡散の移動量や，自ノード以外のノードの関数関数による P の変化量を適用する.

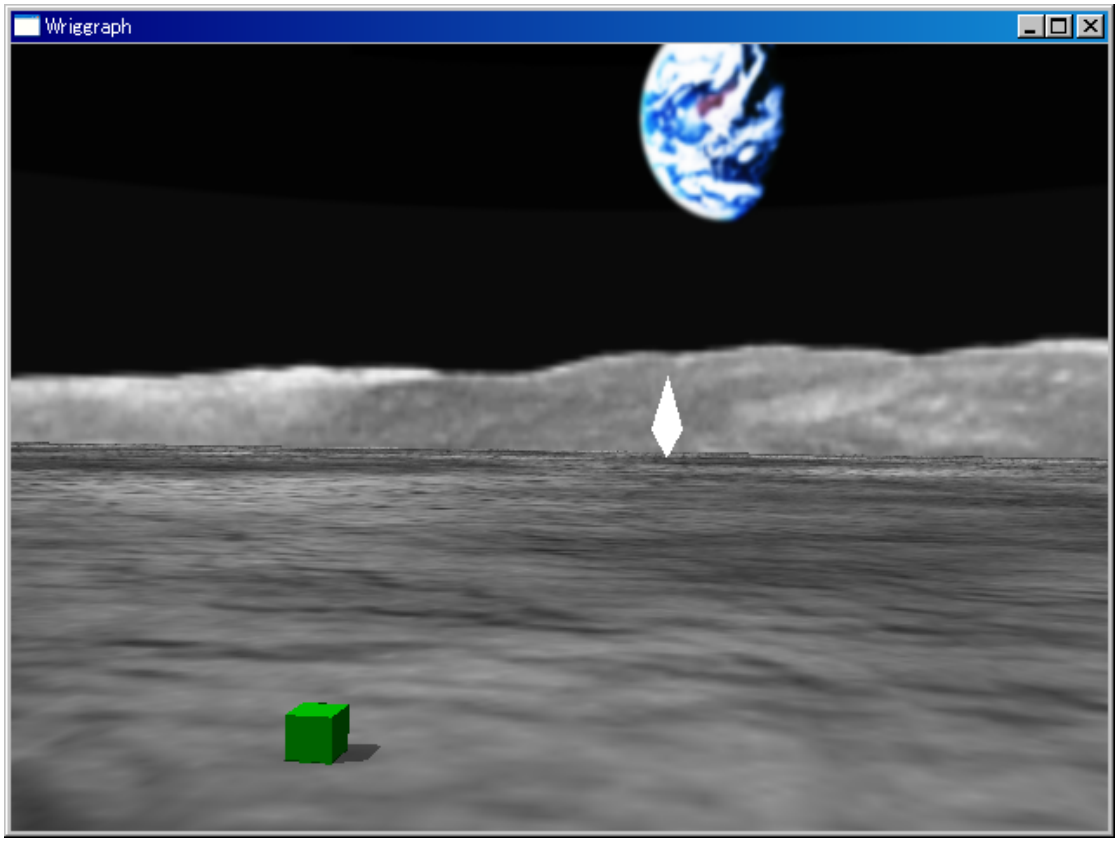


図 4.1: シミュレーション空間.

なお，処理するノードの順番は毎回ランダムである．

表 4.2: シミュレーションで用いたパラメーター.

パラメーター名	**数値**
空間の広さ $(W \times D \times H)$	$1000 \times 1000 \times 50$
重力加速度	0.01
反発係数	0.8
速度抵抗係数 a	0.03
エッジの最大の長さ l_{max}	8
エッジの最小の長さ l_{min}	3
エッジのばね定数 k	0.02
ノードの質量	1
表示するノードの一辺	1
ノードの最大静止摩擦係数 μ^s_{max}	2
ノードの最小静止摩擦係数 μ^s_{min}	0.2
ノードの最大動摩擦係数 μ^d_{max}	1
ノードの最小動摩擦係数 μ^d_{min}	0.1
化学物質の数 N	10

第5章　人工生命体のデザイン

本章では，シミュレーション空間において設定された課題を解決できる人工生命体をWriggraph を用いてデザインする．このことにより，Wriggraph の有用性を確認する．

ここでは，以下の課題を設定する．

1. 単一のノードから，一定ノード数を持つ形態を発生させる．
2. 形態発生中や発生後にノードを人為的に削除しても，元の形状に再生できる．
3. 設定された空間内の目標座標に向かって移動する．

以上の課題によって，Wriggraph の有用性として以下の点を確認する．

- 1 と 2 により，人工生命体自身の形態を自律的に調整できるか
- 2 により，外部要因に起因するエラーに対する堅牢性があるか
- 3 により，外部の環境情報を認識し，それに伴って構成要素を協調させて，あるマクロな動作が可能であるか

以下に，設定された課題を解決することのできる人工生命体「テトラ」の遺伝子コードを構築していく手順を示す．

5.1　ノードの複製

テトラの形態を発生させるためには，ノードの複製が必要である．ノードを複製するには，TDIVIDE 関数や GDIVIDE 関数を用いる．これらの関数の引数 v により得られる化学物質の量 $P_{I(v)}$ が 1 以上であればノードは複製される．このため，$P_{I(v)}$ の値を増加させるコードが必要になる．ノードを複製する簡単なコードをリスト 5.1 に記す．

リスト 5.1: 単純なノード複製

```
1 PROG2 (
2         ADDPAR ( i x1 )
3         GDIVIDE ( i )
4 )
```

コード中の「(」,「)」は見易さのためのものであり，シミュレーターが読み込む際に無視されるので実際にはなくとも支障はない．i や x1 はコードの説明のための変数であり，実際の遺伝子コードにおいては実数を当てはめる．引数 i で参照できる P の値を P[i] とする (P[i]$= P_{I(i)}$)．2 行目で，P[i] に適当な値 x1 を加算している．3 行目で，P[i] によ

るノード複製を行っている．この場合，P[i] がノードを複製させる因子であるといえる．この値が 1 以上であればノードの複製を行う．つまり，ノード複製因子 P[i] に値 x を足していき，P[i]<1 の間はノードの複製は起きない．そして，P[i] ≥ 1 になったところでノードが複製される．このとき，ノード複製因子 P[i] は 0 に戻される．このコードは単位時間ごとにすべてのノードで実行されるので，新しく複製されたノードでも同様のことが起きる．

以上のようにしてノードを複製することができる．このコードでは，すべてのノードが複製という同様の機能を持ち，その結果ノードの複製が無制限に行われることになる．それでは一定のノード数を維持していることにならない．これは，ノード数の調整を伴わない単純な複製であるといえる．図 5.1 に，リスト 5.1 を用いて表現できる人工生命体の例を示す．ここでは，i = 0，x1 = 0.5 を用いた．

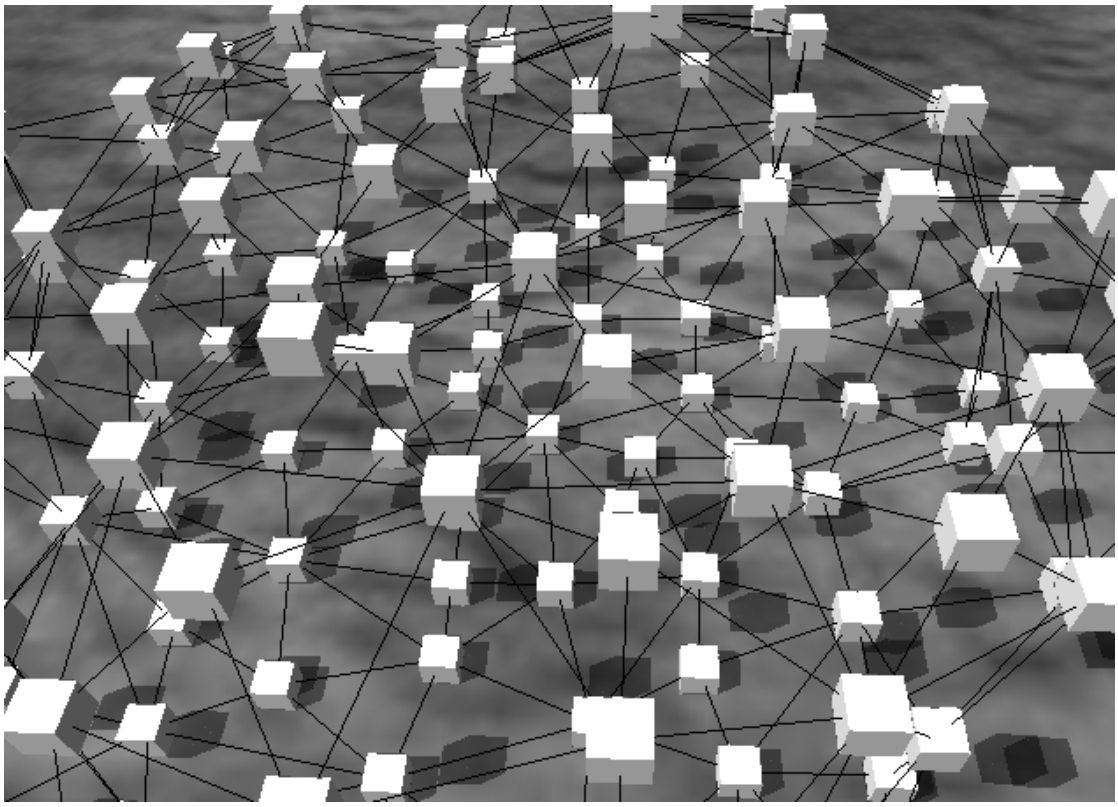


図 5.1: 単純なノード複製による人工生命体．すべてのノードが同様に複製するので，全体では袋状の形態となる．

そこで，次のような二種類の機能を持つノードを考える．まず一つ目は，ノードの複製因子を生産し続けるノードである．そして二つ目は，ノードの複製因子を生産せず，複製因子を生産するノードから拡散した複製因子を得て，複製因子が十分な量に達したとき複製を行うノードである．

二つ目の機能を持つノードの増加にしたがって，個々のノードに拡散する複製因子の量が不足するようになり，二つ目の機能を持つノードが十分な複製因子を得ることが出来な

くなる．その結果，全体のノード数が一定を保つようになる．また，二つ目の機能を持つノードを人為的に削除すれば，個々のノードに拡散する複製因子が増加し，その結果ノードが複製され，全体のノード数が元に戻るという再生作用も期待できる．

一つ目の機能を示すコード例をリスト 5.2 に記す．

リスト 5.2: 複製因子の生産

```
1 PROG2 (
2         SETD ( j d )
3         ADDPAR ( j x1 )
4 )
```

ここで用いる複製因子は，添え字 j で参照できる P[j] とする．2 行目で，複製因子 P[j] に対する拡散係数 D[j] (D[j]= $D_{I(j)}$) に適当な値 d を設定している．これにより，複製因子 P[j] がノード間で拡散するようになる．そして 3 行目で複製因子を増加させている．

次に，二つ目の機能を示すコード例をリスト 5.3 に記す．

リスト 5.3: 複製因子によるノード複製

```
1 PROG3 (
2         SETD ( j d )
3         SUBPAR ( j y1 )
4         GDIVIDE ( j )
5 )
```

一つ目の機能と同じく，2 行目で複製因子に対する拡散係数に d を設定している．そして 3 行目で，SUBPAR 関数により複製因子の値を y1 だけ消費している．この操作により，複製の速度を調整することが出来る．4 行目で複製因子 P[i] が 1 以上であれば複製している．

以上のような二種類の遺伝子コードをデザインすれば，テトラはノード数を一定に保つことが可能である．しかし，シミュレーション空間に最初に配置されるノード数はただひとつであるため，単一の遺伝子コードに以上の二種類の機能を持たせる必要がある．そこで，同一の遺伝子コードを備えたノード同士が相互作用により機能分化するための仕組みを導入する．

5.2 反応拡散系の導入

反応拡散系 [30] とは，化学反応と分子の拡散を組み合わせた反応システムのことである．反応拡散系は自然界に見られるさまざまな生物に関わる現象をよく再現する．特に，生物の形態形成や体表面の模様について，一様な構造から複雑な構造が生じる空間的パターンの研究によく用いられる．そのような空間パターンはチューリングパターン [29] とも呼ばれる．チューリングパターンの例を図 5.2 に示す．

反応拡散系を導入することで，複数ノード間の相互作用によりノードの機能分化を発生させることが出来る．ここでは，以下の反応拡散方程式を用いる．

$$\frac{du}{dt} = f(u,v) + D_u \bigtriangledown^2 u \tag{5.1}$$

$$\frac{dv}{dt} = g(u,v) + D_v \bigtriangledown^2 v \tag{5.2}$$

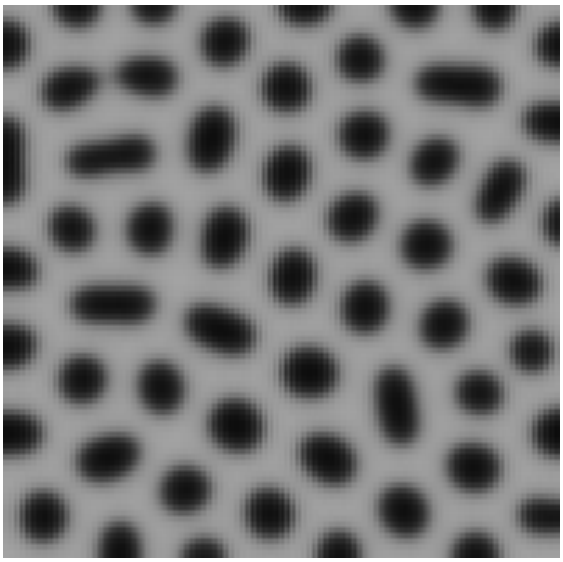


図 5.2: チューリングパターンの例．([31] により生成)

両式の右辺第 1 項を反応項，第 2 項を拡散項と呼ぶ．ある遺伝子コードの機能を発現させる活性因子を u，その活性因子を抑制する因子を v とする．上の式は 2 変数の化学振動子が拡散を通じて空間的に連結されている様子を表しており，反応項に適当な関数を当てはめることにより，v と u の量に関する空間的なパターンが発生する．上の式を遺伝子コードで表すとリスト 5.4 のようになる．ここでは，

$$f(u,v) = au - bv$$

$$g(u,v) = cu - dv$$

を用いた．

リスト 5.4: 反応拡散系

```
1  PROG2 (
2          PROG2  (
3                   SETD ( u du )
4                   SETD ( v dv )
5          )
6          PROG2  (
7                   ADDPAR (
8                            u
9                            SUB (
10                                   MUL ( a GETP ( u ) )
```

```
11                                      MUL ( b GETP ( v ) )
12                              )
13                      )
14                      ADDPAR (
15                              v
16                              SUB (
17                                      MUL ( c GETP ( u ) )
18                                      MUL ( d GETP ( v ) )
19                              )
20                      )
21              )
22  )
```

リスト 5.4 において，活性因子に関する P の添え字を u，抑制因子に関する P の添え字を v とすると，活性因子 u の量は P[u]，抑制因子 v の量は P[v] で参照できる．また，式 5.1，5.2 の拡散項における u と v の拡散係数 D_u, D_v に，それぞれ D[u]，D[v] を対応させる．

リスト 5.4 では，3 行目と 4 行目でそれぞれの拡散係数に値を設定している．7 行目から 13 行目にかけて式 5.1 における反応項を，14 行目から 20 行目にかけて式 5.2 の反応項を表現している．

この遺伝子コードにより，複数ノード間で P[u] の値に不均一性が現れる．リスト 5.4 の実装例を図 5.3 に示す．ここでは，活性因子の量を赤色の強さで，抑制因子の量を青色の強さで表現している．ノード間において因子の空間的パターンが形成されているのがわかる．

5.3 ノード複製の自律的調整

リスト 5.1，リスト 5.2，リスト 5.3，リスト 5.4 を用いて，ノード数を自律的に調整できる遺伝子コードを構成する．遺伝子コードは以下のような部分から構成される．

一つ目は，初期値の設定である．ここでは，反応拡散系における拡散項や複製因子の拡散係数，反応項における各因子の初期値を設定する．

二つ目は，ノードの単純な複製である．反応拡散系は，十分な数のノードが空間に存在しなければ因子の空間パターンが生じない．そのため，発生の初期ではリスト 5.1 で示した単純な複製方法を用いてある程度ノード数を増やしておく．この単純な複製方法に利用する複製因子を複製因子 1 とする．

三つ目は，因子の空間的パターン発現である．リスト 5.4 で示した反応拡散系を用い，活性因子の量が多いノードと少ないノードにノードを分化させる．

四つ目は，活性因子を用いたノードの複製である．リスト 5.2，5.3 で示した複製方法を利用する．このとき，活性因子の量が多いノードのみが，複製因子を生産するように構成する．このとき利用する複製因子を複製因子 2 とする．また，複製因子 2 が複製因子 1 の生産を抑制することで，単純なノード複製の機能を阻害する．これにより，複製方法を切り替える．

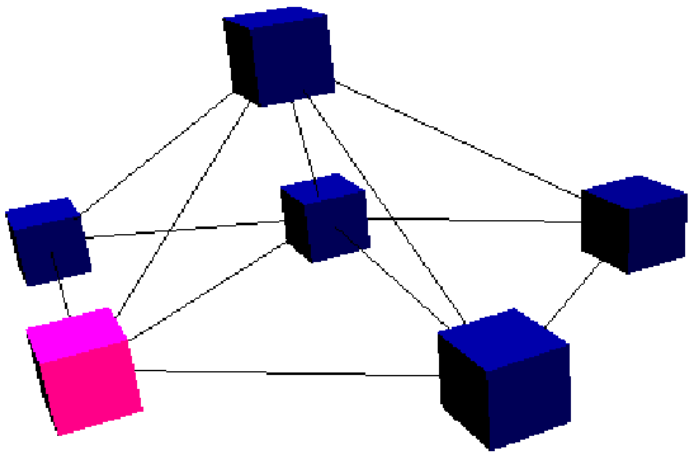


図 5.3: ノード間における因子の空間的パターンの例.

以下に，遺伝子コードの例をリスト 5.5 に示す.

リスト 5.5: ノード複製の自己調整

```
1  PROG3 (
2          < 単純なノード複製 >
3          PROG3 (
4                  < 複製因子２が少なければ複製因子１を増加 >
5                  IFT (    LT ( GETP ( j ) t1 )
6                           ADDPAR ( i x1 )     )
7                  < 複製因子２により複製因子１を抑制 >
8                  SUBPAR ( i GETP ( j ) )
9                  < 複製因子１が１以上ならノード複製 >
10                 GDIVIDE ( i )
11         )
12         < 反応拡散系 >
13         PROG2 (
14                 < 活性因子，抑制因子の初期値設定 >
15                 PROG3 (
16                         IFT (    NOT ( GETP ( u ) )
```

```
17                                      ADDPAR ( u u0 )
18                              )
19                              IFT (   NOT ( GETP ( v ) )
20                                      ADDPAR ( v v0 )
21                              )
22                              < 活性因子, 抑制因子の拡散係数を設定 >
23                              PROG2 (
24                                      SETD ( u du )
25                                      SETD ( v dv )
26                              )
27                      )
28                      < 計算部 >
29                      PROG2 (
30                              ADDPAR (
31                                      u
32                                      SUB (
33                                              MUL ( a GETP ( u ) )
34                                              MUL ( b GETP ( v ) )
35                                      )
36                              )
37                              ADDPAR (
38                                      v
39                                      SUB (
40                                              MUL ( c GETP ( u ) )
41                                              MUL ( d GETP ( v ) )
42                                      )
43                              )
44                      )
45              )
46              < 自己調整ノード複製 >
47              PROG3 (
48                      < 複製因子２の拡散係数を設定 >
49                      SETD ( j dj )
50                      < 活性因子が複製因子２を生成 >
51                      ADDPAR ( SUBPAR ( j y1 ) GETP ( u ) )
52                      < 複製因子２によるノード複製 >
53                      IFT (   LT ( GETP ( u ) t2 )
54                              TDIVIDE ( j )
55                      )
56              )
57  )
```

<, > で囲まれた部分はコメント文であり，遺伝子コードの動作には影響を与えない．

遺伝子コードで用いる因子として，活性因子を P[u]，抑制因子を P[v]，複製因子 1 を P[i]，複製因子 2 を P[j] とおく．以下に，遺伝子コードを構成している部分について述べる．

一つ目は，初期値の設定である．16 行目から 21 行目において，反応拡散系の活性因子 P[u] の値が 0 であれば初期値として u0 を設定する．抑制因子 P[v] の値が 0 であれば初期値として v0 を設定している．24 行目と 25 行目で活性因子 P[u] の拡散係数として du を，抑制因子 P[v] の拡散係数として dv を設定している．49 行目では複製因子 2(P[j]) の拡散係数として dj を設定している．

二つ目は，3 行目から 11 行目までにおけるノードの単純な複製である．

三つ目は，29 行目から 44 行目までにおける反応拡散系の計算である．二つ目の部分で十分なノード数を得た後は，反応拡散系によって活性因子が多いノードとほとんど持たないノードの 2 種類のノードが現れる．

四つ目は，47 行目から 56 行目までにおける P[u] によるノードの複製である．51 行目で，SUBPAR により P[j] の量を適当な値 y2 により減少させている．これは，複製因子 2 の増加量を調整するための操作である．SUBPAR は，第 1 引数の値を返すため，同じ行の ADDPAR は同じ添え字 j を利用できる．それにより，P[j] に P[u] を加算し，複製因子 2 を生成する．このとき，活性因子を持たないノードに複製因子 2 が拡散する．このとき，53，54 行目において，P[u] が適当な閾値 t2 より少ないノードでなければノード複製を行わないようにしている．このため，複製因子 2 の生産のみを行うノードと，複製自体を行うノードの 2 種類に機能分化が生じている．また，5 行目と 6 行目は，P[j] の値が適当な閾値 t1 より少ないときだけ P[i] が増加することを表現する．よって，複製因子 2 を用いるノード複製が開始すれば，複製因子 1 による単純な複製は行われなくなる．

この遺伝子コードの変数に適切な数値を当てはめ，シミュレーションを実行すると，活性因子を持たないノードを人為的に取り除いても，自律的にノードを複製し元の姿に再生することが出来る．

図 5.4 にリスト 5.5 を用いたテトラの形態発生の様子を示す．ノードは複製因子 1 の量を緑色の強さ，活性因子の量を赤色の強さ，抑制因子の量を青色の強さで示している．ノードが一つずつ形態発生する様子がわかる．また，初期のノードは複製因子 1 の量が多いが，活性因子の量が多いノードが現れると複製因子 1 の量が減少しているのがわかる．図 5.4(d) で，ノードの複製が停止する．テトラは正四面体の形状を持つことがわかる．

図 5.5 にリスト 5.5 を用いたノードの再生の様子を示す．図 5.5 の (a) において，ノード数が複製因子 2 による複製が起きない程度に増加して安定状態となっている．ここで，活性因子の少ないノード（円で囲まれた青色のノード）を削除すると図 (b) のように上にあるノードが地面に着き，図 (c) となる．活性因子の多いノード（赤色のノード）が生産する複製因子 2 の量が青色のノード間で十分に拡散し，その結果複製因子 2 によるノードの複製が起こり，結果図 (d) のように元の形状に再生する．

これらの図では，変数に a = 0.9, b = 0.6, c = 0.7, d = 0.3, t1 = 0.0001, t2 = 0.5, du = 0, dv = 0.25, u0 = 0.1, v0 = 0.1, i = 0, j = 0.2, dj = 0.5, u = 0.3, v = 0.4, x1 = 0.09, y1 = 0.3 を用いている．

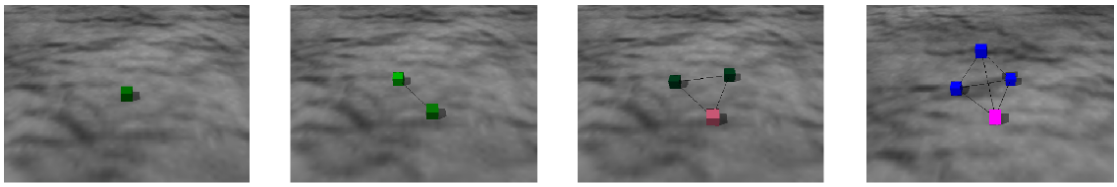


図 5.4: 形態発生の様子.

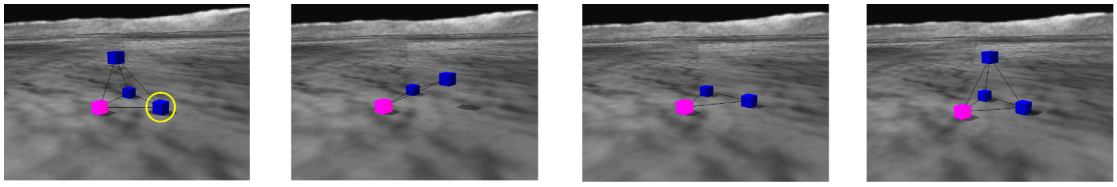


図 5.5: ノードが再生する様子.

5.4 運動

Wriggraph では，人工生命体はエッジを伸縮させることで運動することができる．エッジの長さは，ELENGTH 関数で指定できる．エッジを伸縮させる遺伝子コード例をリスト 5.6 に示す.

リスト 5.6: エッジの伸縮

```
1 PROG3 (
2          ADDPAR ( e x2 )
3          ELENGTH ( e )
4          IMPULSE ( e 0 )
5 )
```

エッジの長さとして P[e] を用いる．2 行目で，P[e] に適当な値 x2 を加算している．3 行目で P[e] の値をエッジの長さとして指定している．3 行目では IMPULSE 関数を用いて，P[e] の値が 1 になれば P[e] を 0 に再設定している．他のノードのエッジの長さに影響を与えないため第 2 引数に 0 を指定して，他のノードの P[e] に値を加算しないようにしている.

この遺伝子コードを実行すると，エッジの長さが最小値から徐々に伸び，一定以上の長さになった瞬間，最小値に戻るという伸縮運動を繰り返す.

5.5　移動する人工生命体の構築

5.4 節で示したエッジの伸縮運動を用いて，テトラを地面の上で移動させる．伸縮運動が人工生命体を構成するすべてのノードで生じた場合，生命体の全体が伸縮するだけで効率的な移動は出来ない．そこで，伸縮運動する部分を最小限にし，他の部分が運動部分に牽引されるような移動を考える．

ノードを運動部分と非運動部分に機能分化させるため，5.2 節で記した反応拡散系を用いる．5.3 節と同様に，反応拡散系の活性因子の量が多いノードに着目し，運動部分として機能分化させる．遺伝子コードの例をリスト 5.7 に示す．

リスト 5.7: 移動機能の分化

```
1  PROG3 (
2          < 活性因子の値をエッジ因子に足す >
3          ELENGTH (
4                  ADDPAR (
5                          e
6                          GETP ( u )
7                  )
8          )
9          < 活性因子が低ければ，エッジ因子の値を一定に保つ >
10         IFTE (  LT ( GETP ( u ) t3 )
11                 SETP ( e l )
12         )
13         IMPULSE (
14                 e
15                 0
16         )
17 )
```

リスト 5.7 では，P[e] をエッジの長さに割り当てるエッジ因子とする．また，P[u] は拡散反応系の活性因子の量を示しているとする．

リスト 5.7 の 4 行目から 7 行目において，P[u] の量を P[e] に加算している．10 行目から 12 行目において，P[u] の量が適当な閾値 t3 より少ないノードにおいては，エッジの長さを適当な値 l に保つようにし，人工生命体の運動部分と非運動部分を機能分化させている．13 行目から 15 行目では，P[e] が 1 なら 0 に値を再設定しており，運動部分が伸縮運動を行うようにしている．

上の遺伝子コードと，5.3 節で示した遺伝子コードを組み合わせたコード例をリスト 5.8 に示す．

リスト 5.8: ノード複製の自己調整と運動機能の統合

```
1  PROG3 (
2          < 複製制御 >
3          PROG2 (
```

```
                < 単純複製 >
                PROG3 (
                        IFT (   LT ( GETP ( j ) t1 )
                                ADDPAR ( i x1 ) )
                        SUBPAR ( i GETP ( j ) )
                        GDIVIDE ( i )
                )
                < 調整複製 >
                PROG3 (
                       < 複製因子２の拡散係数を固定 >
                       SETD ( j dj )
                       < 活性因子が複製因子２を生成 >
                       ADDPAR ( SUBPAR ( j y1 ) GETP ( u ) )
                       IFT (     LT ( GETP ( u ) t2 )
                                 TDIVIDE ( j )
                         )
                )
        )
        < 反応拡散系 >
        PROG2 (
                < 初期値設定 >
                PROG3 (
                        IFT (   NOT ( GETP ( u ) )
                                ADDPAR ( u u0 )
                        )
                        IFT (   NOT ( GETP ( v ) )
                                ADDPAR ( v v0 )
                        )
                        PROG2 (
                                < 活性因子の拡散係数を固定 >
                                SETD ( u du )
                                SETD ( v dv )
                        )
                )
                < 計算部 >
                PROG2 (
                        ADDPAR (
                                u
                                SUB (
                                        MUL ( a GETP ( u ) )
                                        MUL ( b GETP ( v ) )
```

```
45                                      )
46                              )
47                              ADDPAR (
48                                      v
49                                      SUB (
50                                              MUL ( c GETP ( u ) )
51                                              MUL ( d GETP ( v ) )
52                                      )
53                              )
54                      )
55              )
56              < 運動部 >
57              PROG3 (
58                      < 活性因子の値をエッジ因子に足す >
59                      ELENGTH (
60                              ADDPAR (
61                                      e
62                                      GETP ( u )
63                              )
64                      )
65                      < 活性因子が低ければ，エッジ因子の値を一定に保つ >
66                      IFT (   LT ( GETP ( u ) t3 )
67                              SETP ( e 1 )
68                      )
69                      IMPULSE (
70                              e
71                              0
72                      )
73              )
74  )
```

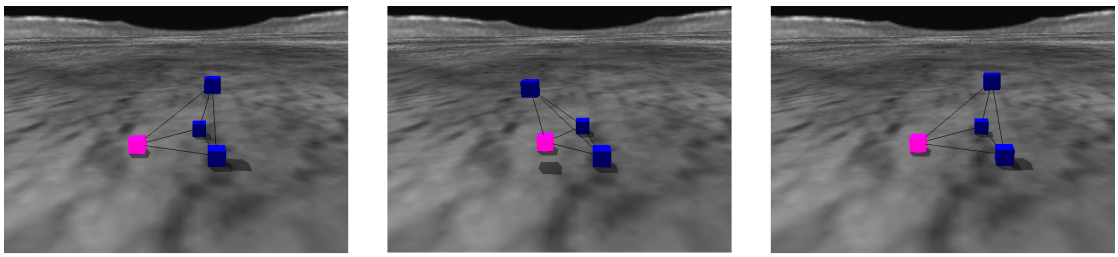
3 行目から 55 行目は，5.3 節までで説明したノードの複製制御に関するコードである．

57 行目から 73 行目は，5.5 節で説明した移動制御に関するコードである．活性因子が多い場合，活性因子をエッジ因子に加算してエッジの長さを伸縮させ，地面と接しているノードを動かすことでテトラを移動させる．

リスト 5.8 で表現できる，移動するテトラの様子を図 5.6 に示す．ノードの色は，赤色の強さが活性因子の量を，青色の強さは抑制因子の量を示している．図 5.6(a) は運動の前段階の各ノードの位置を示している．活性因子の量が多い赤色のノードと抑制因子の量が多い青色のノード二つが地面に接しており，抑制因子の量が多い青色のノード一つがそれら三つのノードに支えられている．図 5.6(b) で，活性因子の多い赤色のノードがエッジを縮ませ，その反動で自分を前に移動させている．図 5.6(c) では赤色のノードがエッジのばねの働きで前にある二つの青いノードを押している．この動作が繰り返されることにより，

テトラは移動する．

この例では，変数に t3 = 0.5，e = 0.8，l = 0.5 を用いている．そのほかの変数は，図 5.5 で利用した変数と同様の数値を用いている．



（a）　（b）　（c）

図 5.6: 移動する様子．

5.6　目的地に移動する人工生命体の構築

遺伝子コードには，あらかじめ設定しておいた目標座標との近さを出力する SMELL 関数が用意されている．この関数は，ノードが目標座標に近づくほど 1 に近い値を出力する．この関数を用いて，テトラを目標座標に向かって移動させる．

図 5.6 におけるテトラは，駆動部分となるひとつの赤色のノードと，地面に接していないノード，そして駆動ノードに押される二つの地面に接しているノードで成り立っている．駆動ノードに押される二つのノードには地面との間に一定の摩擦力がかかっている．左右の二つのノードで摩擦係数に違いがあると，摩擦係数が高いノードはあまり移動せず，低いノードのほうがよく移動するので，テトラは摩擦係数が高いノードのほうに回り込みながら移動することになる．このため，目標座標に近いノードの摩擦係数が高ければ，テトラは目標座標に向かって移動できる．このことを実現するためには，ひとつのノードがもうひとつのノードよりも目標座標に近いのか遠いのかを判断する必要がある．

そこで，次に述べる方法を用いてその判断を行う．まず，SMELL 関数で得られた値を P[s] に代入する．また，P[s] の拡散係数 D[s] に，P[s] が拡散するように適当な値を代入しておく．そして，P[s] の値をノード間で拡散させる．このとき，P[s] の値が高いノードから低いノードへ拡散する．その結果，拡散した後の P[s] の値と再度 SMELL で得られた値を比べると，拡散前に P[s] の値が高かったノード，つまり目標座標に近かったノードでは，P[s] の値が SMELL 関数で得られた値より低くなっている．このようなノードの摩擦係数を高くし，そのほかのノードの摩擦係数を低くすれば，テトラは目標座標に向かって移動する．リスト 5.9 に遺伝子コードの例を示す．

リスト 5.9: 目的地移動のための摩擦係数設定

```
1 PROG3 （
2         IFTE （  LT （ GETP （ s ） SMELL ）
3                  < こちら側は目標に近い側 >
```

```
4                   PROG2 (
5                           SETP ( dc q1 )
6                           SETP ( sc q2 )
7                   )
8                   < こちら側は目標に遠い側 >
9                   PROG2 (
10                          SETP ( dc r1 )
11                          SETP ( sc r2 )
12                  )
13          )
14          PROG2 (
15                  SETP ( s SMELL )
16                  SETD ( s ds )
17          )
18          PROG2 (
19                  DYNFRIC ( dc )
20                  DIFFRIC ( sc )
21          )
22  )
```

リスト 5.9 を構成する部分について以下に述べる．

一つ目は，値の設定である．15 行目で P[s] に SMELL 関数の値を設定している．16 行目で P[s] の拡散係数 D[s] に適当な値 ds を設定している．19 行目で関数 DYNFRIC に動摩擦係数に関して P[dc] の値を参照するように設定し，20 行目で関数 DIFFRIC により静止摩擦係数に関する値として P[sc] を参照するように設定する．

二つ目は，値の拡散後における値の比較である．2 行目で，拡散後の P[s] と SMELL 関数が返す値を比較する．P[s] の値が低ければそのノードは目標により近いほうであるとして，5 行目 6 行目で P[dc] と P[sc] にそれぞれ適当な値 q1,q2 を設定して摩擦係数を設定している．拡散後の P[s] が SMELL 関数の値より高ければ，q1,q2 より低い値 r1,r2 を利用し摩擦係数を低く設定している．

ここで一つ注意点を述べる．SMELL 関数による値の設定の次に，値の拡散が行われ，そして最後に拡散後の値と SMELL 関数の返り値との比較を行うのが通常の処理の流れである．しかし，シミュレーションでは値の拡散という操作が遺伝子コードの実行の前後に行われるため，リスト 5.9 では，拡散後の値と SMELL 関数の返り値との比較のあとで，SMELL 関数による値の設定を行っている．

5.5 節までに示した遺伝子コードに組み込んだ例をリスト 5.10 に示す．

リスト 5.10: 複製制御と運動方向制御とを組み合わせた遺伝子コード

```
1  PROG3 (
2           < 複製制御 >
3           PROG2 (
4                   < 単純複製 >
```

```
5                   PROG3 (
6                           IFT (    LT ( GETP ( j ) t1 )
7                                    ADDPAR ( i x )
8                           )
9                           SUBPAR ( i GETP ( j ) )
10                          GDIVIDE ( i )
11                  )
12                  < 調整複製 >
13                  PROG3 (
14                         < 複製因子２の拡散係数を固定 >
15                         SETD ( j dj )
16                         < 活性因子が複製因子２を生成 >
17                         ADDPAR ( SUBPAR ( j y1 ) GETP ( u ) )
18                         IFT (      LT ( GETP ( u ) t2 )
19                                    TDIVIDE ( j )
20                            )
21                  )
22          )
23          < 反応拡散系 >
24          PROG2 (
25                  < 初期値設定 >
26                  PROG3 (
27                          IFT (    NOT ( GETP ( u ) )
28                                   ADDPAR ( u u0 )
29                          )
30                          IFT (    NOT ( GETP ( v ) )
31                                   ADDPAR ( v v0 )
32                          )
33                          PROG2 (
34                                   < 活性因子の拡散係数を固定 >
35                                   SETD ( u du )
36                                   SETD ( v dv )
37                          )
38                  )
39                  < 計算部 >
40                  PROG2 (
41                          ADDPAR (
42                                   u
43                                   SUB (
44                                            MUL ( a GETP ( u ) )
45                                            MUL ( b GETP ( v ) )
```

```
46                                      )
47                              )
48                              ADDPAR (
49                                      v
50                                      SUB (
51                                              MUL ( c GETP ( u ) )
52                                              MUL ( d GETP ( v ) )
53                                      )
54                              )
55                      )
56              )
57              < 運動部 >
58              PROG2 (
59                      < 駆動部 >
60                      PROG3 (
61                              < 活性因子の値をエッジ因子に足す >
62                              ELENGTH (
63                                      ADDPAR (
64                                              e
65                                              GETP ( u )
66                                      )
67                              )
68                              < 活性因子が低ければ，エッジ因子の値を一定に保つ
    >
69                              IFTE (  LT ( GETP ( u ) t3 )
70                                      SETP ( e l )
71                              < 活性因子が高ければ，摩擦係数に高い値を設定 >
72                                      PROG2 (
73                                              SETP ( dc o1 )
74                                              SETP ( sc o2 )
75                                      )
76                              )
77                              IMPULSE (
78                                      e
79                                      0
80                              )
81                      )
82                      < 方向制御部 >
83                      PROG3 (
84                              IFTE (  LT ( GETP ( s ) SMELL )
85                                      < こちら側は目標に近い側 >
```

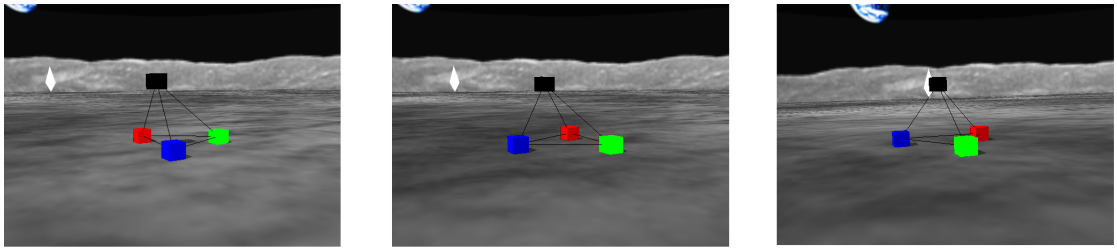
```
86                                      PROG2 (
87                                              SETP ( dc q1 )
88                                              SETP ( sc q2 )
89                                      )
90                                      < こちら側は目標に遠い側 >
91                                      PROG2 (
92                                              SETP ( dc r1 )
93                                              SETP ( sc r2 )
94                                      )
95                              )
96                              PROG2 (
97                                      SETP ( s SMELL )
98                                      SETD ( s ds )
99                              )
100                             PROG2 (
101                                     DYNFRIC ( dc )
102                                     DIFFRIC ( sc )
103                             )
104                     )
105             )
106 )
```

目的地に移動するための方向を制御するコードを 83 行目から 104 行目に挿入している．また，71 行目から 76 行目にかけて，活性因子が高ければ摩擦係数に非常に大きな数値 o1,o2 を設定している．活性因子が多いノードが，テトラ全体を動かす運動を行うため，摩擦係数に高い数値を設定することにより生命体がよく移動できる．

リスト 5.10 で表現できるテトラが移動方向を変えている様子を図 5.7 に示す．目標座標はノードの初期位置から z 軸方向に 250 後方の座標に設定し，図 5.7 の奥にあるひし形として表示する．活性因子の量は緑色の強さで示している．緑色のノードと反対方向がテトラの移動方向である．摩擦係数の低いノードを青色で，摩擦係数の高いノードを赤色で示している．目標座標に近い奥のノードの摩擦係数が高く設定してあることがわかる．図 5.7 の (a) から (c) を比べると，テトラが目標座標であるひし形の方向に向かって移動方向を変えていることがわかる．この例では，変数に s = 0.6，dc = 1.0，sc = 2.0，o1 = 5.0，o2 = 5.0，q1 = 0.9，q2 = 0.9，r1 = 0.1，r2 = 0.1，を用いている．そのほかの変数は，図 5.6 で利用した変数と同様の数値を用いている．

5.7　活性因子が多いノードの再生

5.6 節までで示した遺伝子コードでは，テトラの発生途中や発生後において活性因子が少ないノードを削除しても，5.3 節で説明したように適切にノード数が回復する．しかし，活性因子が多いノードが削除された場合については考慮していない．



（a）　（b）　（c）

図 5.7: 移動方向を制御する様子.

活性因子が多いノードを削除すると，複製因子 2 を生成するノードがなくなり，活性因子を持たないノードが複製因子 1 を再度生成し始める．その結果，複製因子 1 によるノードの複製が行われてノード数が増加し，テトラの形態が不安定になる．

そのため，活性因子が多いノードが削除された場合は，他の活性因子を持たないノードのうちのいずれかが活性因子を生成し，そのノードを活性因子が多いノードとして機能分化させる．その結果，ノードが再生することができる．

そこで，摩擦係数の値により，分化するノードを決定する．5.6 節では，テトラの移動方向を決定するために，活性因子を持たないノードの摩擦係数を目標座標との位置関係で変化させていた．このためノード間で摩擦係数には違いがある．この差異を各ノードの活性因子生成量の差異に反映させることで，活性因子を持たないノードにおける活性因子生成による機能分化の促進を行うことができる．

5.6 節までに示した遺伝子コードに上記の機能を組み込んだコード例をリスト 5.11 に示す．

リスト 5.11: ノード複製機能の改良

```
1  PROG3 (
2          < 複製制御 >
3          PROG2 (
4                  < 単純複製 >
5                  PROG3 (
6                          IFT (    LT ( GETP ( j ) t1 )
7                                   ADDPAR ( i x1 )
8                          )
9                          SUBPAR ( i GETP ( j ) )
10                         GDIVIDE ( i )
11                 )
12                 < 調整複製 >
13                 PROG3 (
14                        < 複製因子 2 の拡散係数を固定 >
15                        SETD ( j dj )
16                        < 活性因子が複製因子 2 を生成 >
```

```
                        ADDPAR ( SUBPAR ( j y1 ) GETP ( u ) )
                        IFT (     LT ( GETP ( u ) t2 )
                                  TDIVIDE ( j )
                           )
                )
        )
        < 反応拡散系 >
        PROG2 (
                < 初期値設定 >
                PROG3 (
                        IFT (   NOT ( GETP ( u ) )
                                ADDPAR ( u u0 )
                        )
                        IFT (   NOT ( GETP ( v ) )
                                ADDPAR ( v v0 )
                        )
                        PROG3 (
                                < 活性因子の拡散係数を固定 >
                                SETD ( u du )
                                SETD ( v dv )
                                < 活性因子増量の補助 >
                                IFT (   AND ( GROUND
                                              AND ( GETP ( dc )
                                                    LT ( GETP ( dc ) t4 )
                                              )
                                        )
                                        ADDPAR ( u GETP ( dc ) )
                                )
                        )
                )
                < 計算部 >
                PROG2 (
                        ADDPAR (
                                u
                                SUB (
                                        MUL ( a GETP ( u ) )
                                        MUL ( b GETP ( v ) )
                                )
                        )
                        ADDPAR (
                                v
```

```
58                                      SUB (
59                                              MUL ( c GETP ( u ) )
60                                              MUL ( d GETP ( v ) )
61                                      )
62                              )
63                      )
64              )
65              < 運動部 >
66              PROG2 (
67                      < 駆動部 >
68                      PROG3 (
69                              < 活性因子の値をエッジ因子に足す >
70                              ELENGTH (
71                                      ADDPAR (
72                                              e
73                                              GETP ( u )
74                                      )
75                              )
76                              < 活性因子が低ければ，エッジ因子の値を一定に保つ
    >
77                              IFTE (  LT ( GETP ( u ) t3 )
78                                      SETP ( e l )
79                              < 活性因子が高ければ，摩擦係数に高い値を設定 >
80                                      PROG2 (
81                                              SETP ( dc o1 )
82                                              SETP ( sc o2 )
83                                      )
84                              )
85                              IMPULSE (
86                                      e
87                                      0
88                              )
89                      )
90                      < 方向制御部 >
91                      PROG2 (
92                  < 活性因子，抑制因子が一定範囲内であれば摩擦係数の設定をする >
93                              IFT (
94                                      AND (
95                                              GROUND
96                                              AND (   GT ( GETP ( u ) t5 )
97                                                      LT ( GETP ( v ) t6 )
```

```
98                                                  )
99                                          )
100                                         PROG2 (
101                                                 IFTE (  LT ( GETP ( s ) SMELL )
102                                                 < こちら側は目標に近い側 >
103                                                         PROG2 (
104                                                                 SETP ( dc q1 )
105                                                                 SETP ( sc q2 )
106                                                         )
107                                                 < こちら側は目標に遠い側 >
108                                                         PROG2 (
109                                                                 SETP ( dc r1 )
110                                                                 SETP ( sc r2 )
111                                                         )
112                                                 )
113                                                 PROG2 (
114                                                         SETP ( s SMELL )
115                                                         SETD ( s ds )
116                                                 )
117                                         )
118                                 )
119                                 PROG2 (
120                                         DYNFRIC ( dc )
121                                         DIFFRIC ( sc )
122                                 )
123                         )
124                 )
125 )
```

5.6 節では，ノードが目標座標より遠い場合，摩擦係数を低く設定した．このノードにおける活性因子を増量させて機能分化させると，移動の主体となるノード（図 5.6 においては赤色のノード）が目標座標の逆側に位置することになり，目標座標の方向を向いて移動しやすくなる．よって，38 行目から 44 行目において，ノードが地面に接地しており，動摩擦係数 `P[dc]` が 0 ではなく，なおかつ適当な閾値 `t4` より低ければ活性因子に値を加算するようにしている．

また，90 行目から 96 行目にかけて，活性因子 `u` が適当な閾値 `t5` より大きく，かつ抑制因子が適当な閾値 `t6` より小さい場合であれば，摩擦係数の設定を行うという条件を追加している．この条件がない場合，シミュレーションの最初に空間に設置されるノードはすぐに摩擦係数を設定する．すると，その時点から活性因子に影響が及ぶことになる．このため機能分化が早すぎる段階で生じ，正常な形態発生が妨げられる．それに対して，ノード数が増え，両因子がある程度まで生成され拡散している時点は安定状態であり，摩擦係数による影響が吸収されうる．このときまで摩擦係数の設定は控える必要がある．そして，

このときに活性因子が多いノードが削除された場合，摩擦係数による活性因子の生成量の増量は機能分化に十分な影響を与えることができる．

図 5.8 に，リスト 5.11 を用いた活性因子の多いノードについての再生の様子を示す．それぞれのノードにおいて，活性因子の量を赤色の強さ，抑制因子の量を青色の強さ，複製因子 1 の量を緑色の強さで示している．図 (a) において，活性因子の量が多いノード (円で囲まれた赤色のノード) を削除すると，上に乗っていたノードが地面に着き図 (c) のようになる．図 (d) では，活性因子の量が多いノードが削除されたため，複製因子 2 が生産されなくなる．その結果，各ノードにおける複製因子 2 による複製因子 1 の生産が抑制されなくなり，複製因子 1 の量が増加する．そのためノードの色は緑色が強くなる．それと同時に，他のノードより目標座標から遠い手前側のノードで活性因子が生産され始める．そのため，図 (e) の手前側のノードの色は赤色が現れている．図 (f) では，手前側のノードは完全に活性因子を生産するノードとして分化している．図 (g) では，複製因子 2 が生産されて他のノードに拡散するため，他のノードの複製因子 1 の生産が抑制されて緑色が消え，その代わりに抑制因子の量を示す青色が強くなっている．図 (h) において，複製因子 2 によるノードの複製が行われ，結果的に元の姿に再生している．この例では，変数に `t4 = 0.5`，`t5 = 0.5`，`t6 = 0.5` を用い．他の変数については図 5.7 で利用した変数と同様の数値を用いている．

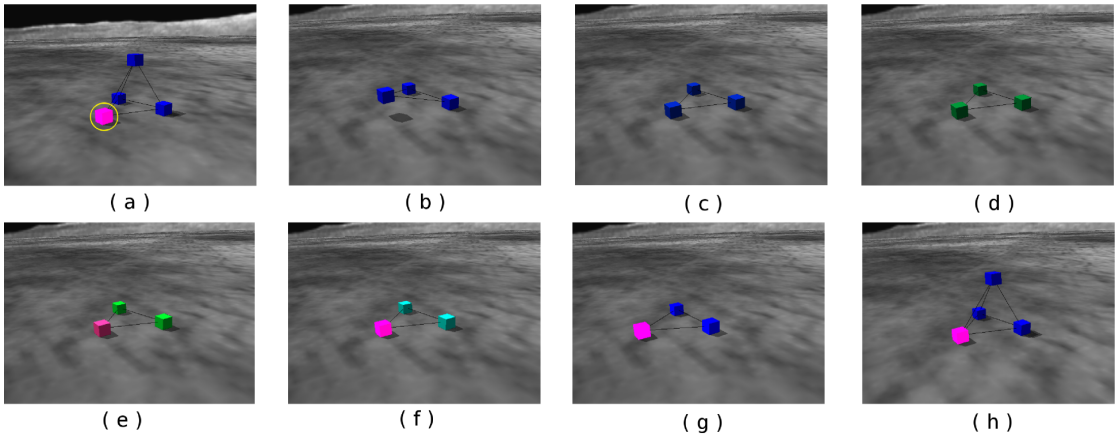


図 5.8: 活性因子量が多いノードについての再生.

5.8　その他の人工生命体

Wriggraph を用いてデザインできる他の人工生命体を以下に述べる．図 5.9 は一定数のノードを持ち，袋状に発生する．図 5.10 のように，目標座標に向かって転がるように移動することができる．リスト 5.12 にこの人工生命を表現する遺伝子コードを示す.

図 5.11 は一定数のノードを持ち，筒状に発生する．リスト 5.13 にこの人工生命を表現する遺伝子コードを示す.

図 5.12 は一定数のノードを持ち，放射状に発生する．リスト 5.14 にこの人工生命を表現する遺伝子コードを示す.

リスト 5.12: 袋状人工生命体を表現する遺伝子コード

```
PROG3 (
       PROG2 (
              ELENGTH ( 0.2 )
              PROG3 (
                     DIFFRIC ( 0.6 )
                     DYNFRIC ( 0.6 )
                     SETP ( 0.6 0.5 )
              )
       )
       PROG3 (
                 GDIVIDE ( 0 )
                 IFT (
                          NOT GETP ( 0 )
                          ADDDIF ( 0 1 )
                     )
                 IFT (
                          LT ( GETD ( 0 ) 0.04 )
                          ADDPAR ( 0 1 )
                      )
               )
       PROG3 (
           IFTE (
                 LT ( GETP ( 0.4 ) SMELL )
                 SUBPAR ( 0.2 1 )
                 IFT ( LT ( GETP ( 0.2 ) 2 )
                        ADDPAR ( 0.2 1 )
                 )
              )
              SETD ( 0.4 1 )
              SETP ( 0.4 SMELL )
       )
)
```

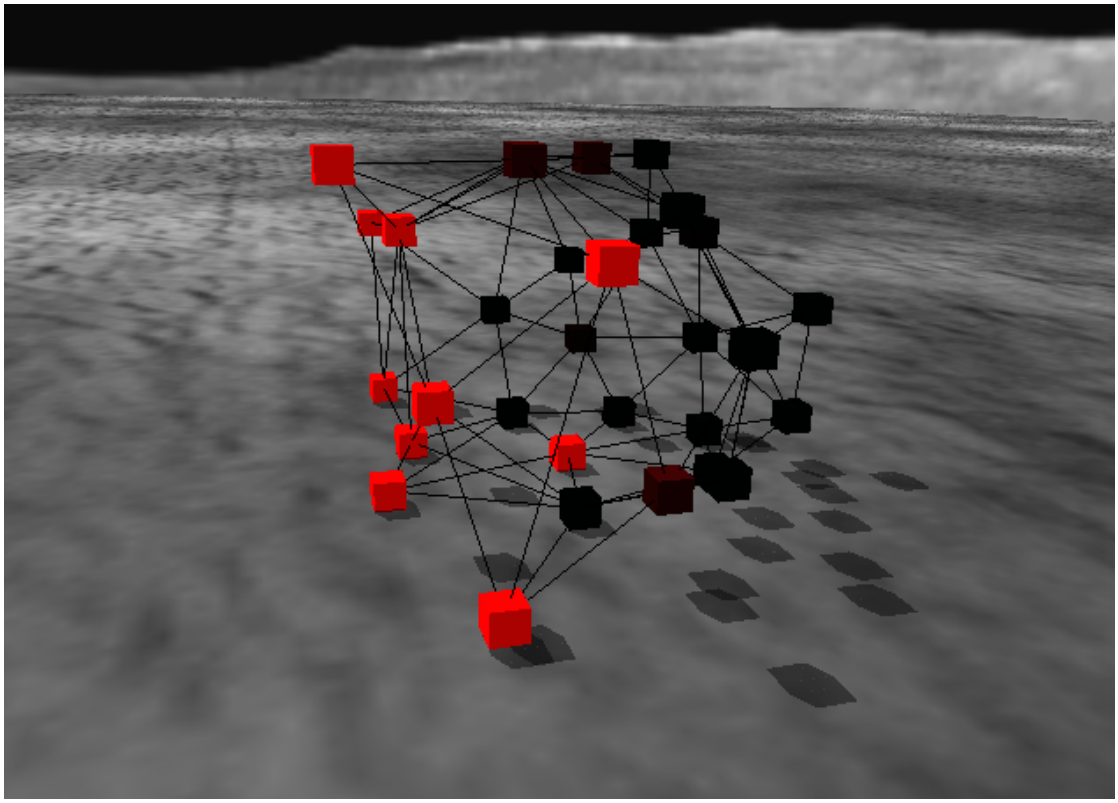
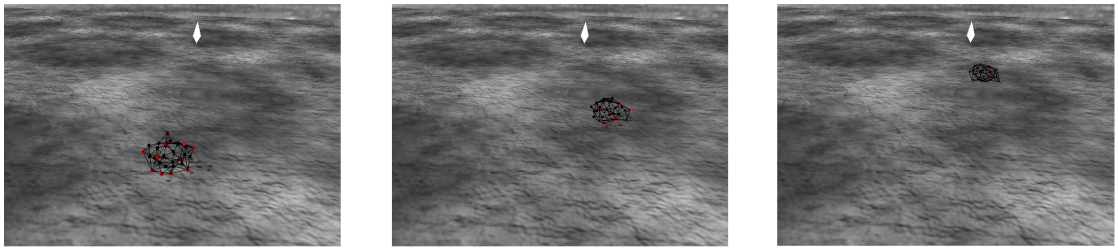


図 5.9: 袋状の人工生命体



(a)　　(b)　　(c)

図 5.10: 人工生命体が転がって移動する様子

リスト 5.13: 筒状人工生命体を表現するの遺伝子コード

```
1  PROG2 (
2           PROG3 (
3                   TDIVIDE (
4                           ADDPAR (
5                                   0
6                                   0.9
7                           )
8                   )
9                   ADDPAR (
10                          0.2
11                          MUL (
12                                  1
13                                  GETP ( 0 )
14                          )
15                  )
16                  SUBPAR (
17                          0
18                          MUL (
19                                  0.4
20                                  GETP ( 0.2 )
21                          )
22                  )
23          )
24           PROG3 (
25                  ADDDIF (
26                          0.2
27                          0.001
28                  )
29                  ELENGTH ( 0.2 )
30                  SETCOLP ( 0 0.9 0.2 )
31           )
32 )
```

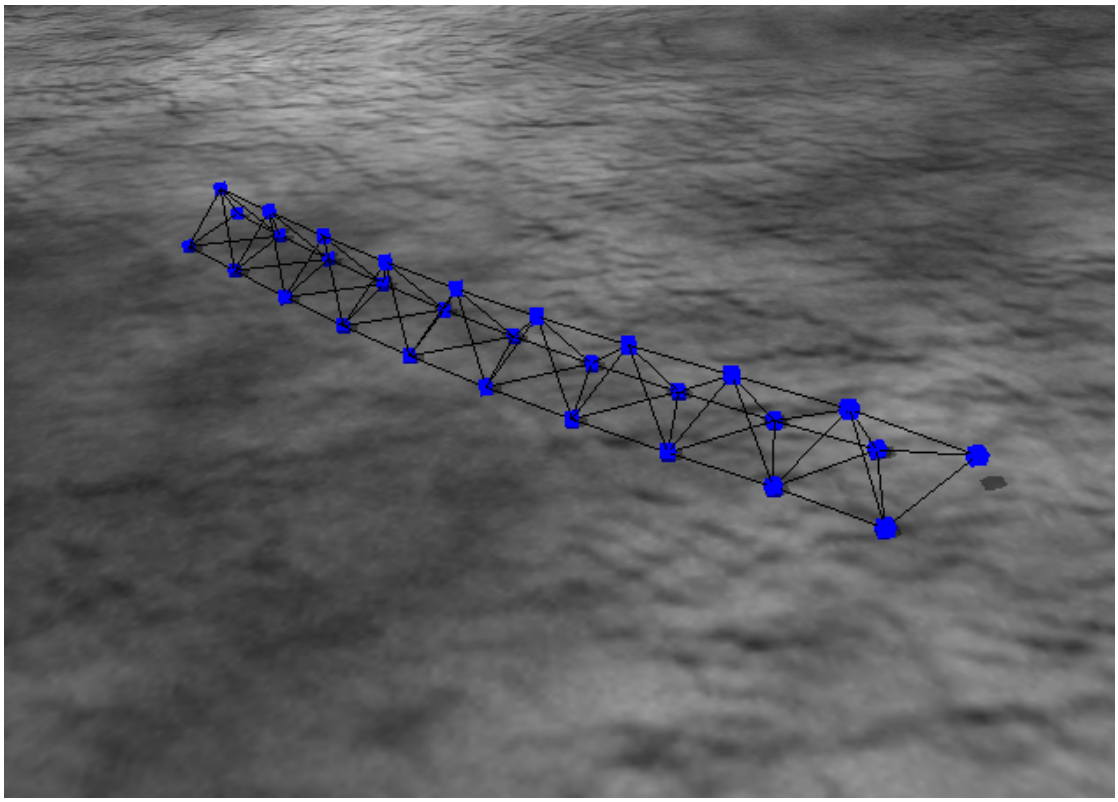


図 5.11: 筒状の人工生命体

リスト 5.14: 放射状人工生命体を表現する遺伝子コード

```
1  PROG2 (
2          PROG3 (
3                  TDIVIDE (
4                          ADDPAR (
5                                  0
6                                  0.9
7                          )
8                  )
9                  ADDPAR (
10                         0.2
11                         MUL (
12                                 0.7
13                                 GETP ( 0 )
14                         )
15                 )
16                 SUBPAR (
17                         0
18                         MUL (
19                                 0.4
20                                 GETP ( 0.2 )
21                         )
22                 )
23         )
24          PROG3 (
25                 ADDDIF (
26                         0.2
27                         0.001
28                 )
29                 ELENGTH ( 0.2 )
30                 SETCOLP ( 0 0.9 0.2 )
31          )
32 )
```

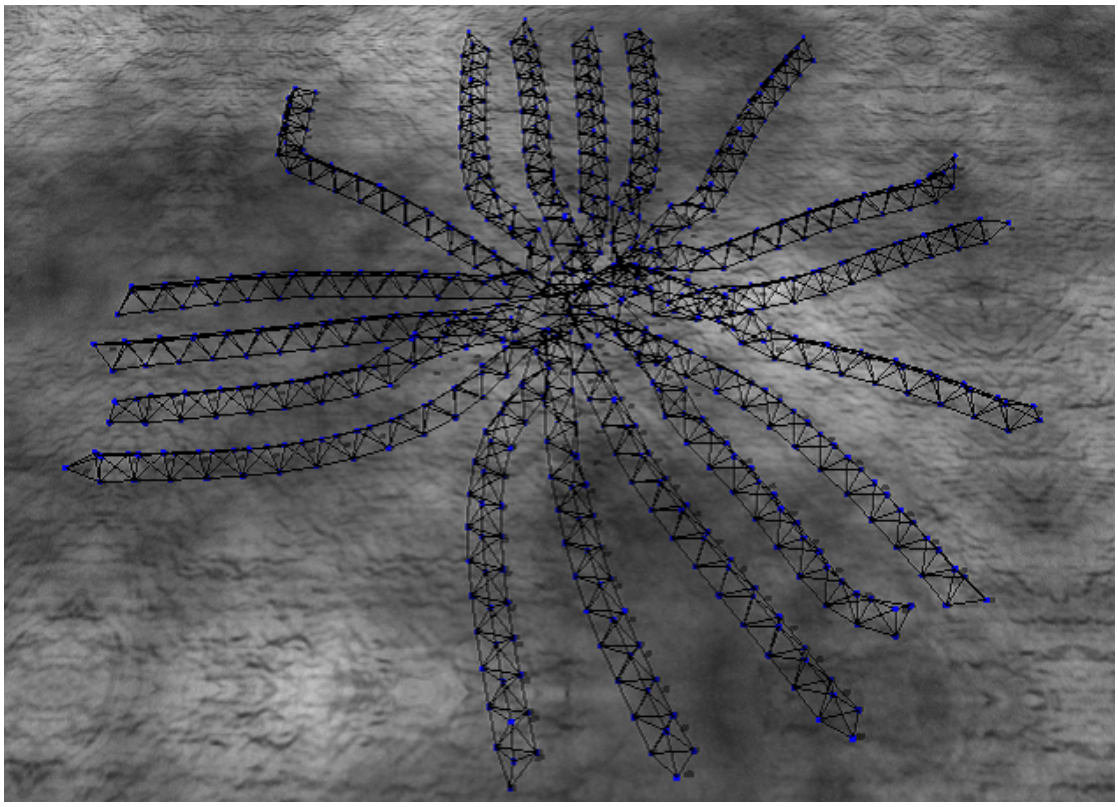


図 5.12: 放射状の人工生命体

第6章　結論と今後の課題

本稿では，embeddedness が高く，形態形成と運動の双方を同一の枠組み内で実現する動力学的グラフモデル Wriggraph を提案した．そして，第 5 章において人工生命体「テトラ」を構築した．テトラは，次のことを実現した．

一つ目は，一つのノードからノードの複製を行い，特定の形状が得られるノード数に達したとき自動的にノードの複製を停止している．このことにより，Wriggraph で表現できる人工生命体は，形態発生の自己制御が可能であることがわかる．

また，二つ目として，形態の発生途中や発生後において，形態を構成しているノードを削除しても，ノードを正常に再生して形態を正常に保つことが出来ている．実世界における人工生命の応用を考えたとき，コンピューター上のシミュレーションと違い，人工生命体の活動中に起こりうる構成要素の故障は常に考慮すべき問題である．人工生命体はいかなる活動中においても，構成要素の故障に対して強力な耐性を示すべきである．Wriggraph で表現できる人工生命体は，形態発生や発生後の活動中において構成要素が削除された場合でも，新しいノードを補充するべき箇所やノードの数を特定できている．このことは他の関連研究では実現していない Wriggraph の強力な利点となる．

三つ目として，運動の制御があげられる．テトラは，目的地に移動するという運動が可能である．移動ベクトルの方向を調整するため，自己の構成要素であるノードの性質を，目標座標への近さという環境情報によって少しずつ変化させている．よって，Wriggraph では環境情報を認識し，ノード同士を協調させてマクロな運動が出来る人工生命体を表現できることがわかる．

これらのことは，すべてノード間の局所的な相互作用によって実現できている．分散化した局所的相互作用がシステムのマクロな挙動を決定しており，この枠組み内において，人工生命の形態発生と運動を決定するシステムの自律的な振る舞いを記述できる．関連研究のように発生の段階と運動の段階を明確に区別し，その二つの段階を人為的に切り替える必要が無い．また，形態についての局所的な調整作用を記述することにより，構成要素のエラーに対する再生という利点を得ることができた．この自己調整的な振る舞いは，システムに生じたエラーに対して自律的に対処する堅牢性を提供するものであり，人為的な対処操作を必要としていない．さらに，人工生命体の活動に必要な構成要素は，すべてモデルの同一枠組み内に埋め込まれている．このため，Wriggraph は embeddedness の度合いが他のモデルに比べて高い．

これらの特徴のため，Wriggraph は関連研究で提案されているモデルと比べて，構造的な現実性や物理的な実装可能性が高いといえる．今後の展開として，極地開発や宇宙開発の現場において，人の直接的操作を必要とせず，構成要素の故障の対して堅牢性を持ち，環境に柔軟に適応しつつ開発を行うような人工物を，Wriggraph で実現していくことが考えられる．

テトラは，発生と運動の制御を同一の原理的枠組み内で記述し，構成要素間の局所的相互作用のみでシステムのマクロな振る舞いを決定できた．しかし，ノードの複製動作についてはまだ物理的現実性が低いため，モデルの改良により複製についての現実性を高めることが必要である．また，テトラは構成要素数が少なく，運動機能も目的地に移動するといった一つの動作だけであるため，遺伝子コードを GP として生成させるといった進化的計算等により，構成要素数がより多く，より複雑な動作を可能とすることが今後の課題である．

発表論文

Koji Sano and Hiroki Sayama, Wriggraph: A kinetic graph model that uniformly describes ontogeny and motility of artificial creatures, Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems, Bloomington, IN, 2006, MIT Press, in press.

参考文献

[1] Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D. 1998. Genetic Programming: An Introduction : On the Automatic Evolution of Computer Programs and Its Applications, dpunkt.verlag and Morgan Kaufmann Publishers, Inc.

[2] Bongard, J. C., and Pfeifer, R. 2001. Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny, in Spector, L. et al (eds.), Proceedings of The Genetic and Evolutionary Computation Conference, GECCO-2001. San Francisco, CA: Morgan Kaufmann publishers, 829–836.

[3] Bongard, J. C., and Pfeifer, R. 2003. Evolving Complete Agents Using Artificial Ontogeny, in Hara, F. and R. Pfeifer, (eds.), Morpho-functional Machines: The New Species (Designing Embodied Intelligence) Springer-Verlag, 237–258

[4] Bongard, J. C. Josh Bongard, http://www.mae.cornell.edu/bongard/

[5] Hornby, G. S., and Pollack, J. B. 2001. Body-Brain Coevolution Using L-systems as a Generative Encoding, Genetic and Evolutionary Computation Conference (GECCO), 868–875

[6] Hornby, G. S., and Pollack, J. B. 2001. Evolving L-Systems to generate virtual creatures, Computers & Graphics 25, 1041–1048

[7] Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., and Kokaji, S. 2005. Automatic Locomotion Design and Experiments for a Modular Robotic System, IEEE/ASME Transactions on Mechatronics, Vol. 10, Issue 3, 314–325

[8] Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., and Kokaji, S. 2004. Distributed adaptive locomotion by a modular robotic system, M-TRAN II. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), 2370–2377.

[9] Komosinski, M., and Ulatowski, Sz. 1999. Framsticks: towards a simulation of a nature-like world, creatures and evolution., Proceedings of 5th European Conference on Artificial Life(ECAL99), September 13-17, Lausanne, Switzerland, Springer-Verlag, 261–265

[10] Komosinski, M. 2000. The World of Framsticks: Simulation, Evolution, Interaction, Proceedings of 2nd International Conference on Virtual Worlds, Paris, France, July 2000., 214–224

[11] Komosinski, M. 2003. The Framsticks system: versatile simulator of 3D agents and their evolution. In: Kybernetes: The International Journal of Systems & Cybernetics, 32 (1/2), 2003. MCB University Press, 156–173

[12] Komosinski, M., and Ulatowski, Sz. 1997-. Framsticks - Artificial Life - 3D evolution and simulation, http://www.frams.alife.pl/

[13] Lindenmayer, A. 1968. Mathematical models for cellular interaction in development, Part I and Part II, Jounal of Theoretical Biology, 18:280-299, 300–315

[14] Lipson, H., and Pollack J. B. 2000. Automatic Design and Manufacture of Artificial Lifeforms, Nature 406, 974–978.

[15] Lipson, H., and Pollack J. B., 2000. The GOLEM Project, http://demo.cs.brandeis.edu/golem/

[16] MTRAN2, http://unit.aist.go.jp/is/dsysd/mtran/

[17] O'Kelly, M. J. T., and Hsiao, K. 2004. Evolving Simulated Mutually Perceptive Creatures for Combat, Ninth International Conference on the Simulation and Synthesis of Living Systems(ALife), 113–118

[18] O'Kelly, M. J. T., BubbleGene, http://www.mit.edu/~mokelly/bubblegene/index.html

[19] Salzberg, C., and Sayama, H. 2004. Heredity, complexity and surprise: Embedded self-replication and evolution in CA, Cellular Automata: Proceedings of the Sixth International conference on Cellular Automata for Research and Industry (ACRI 2004), P.M.A. Sloot, B. Chopard, and A.G. Hoekstra, eds., 161–171

[20] Sims, K. 1994. Evolving Virtual Creatures, Siggraph '94 Proceedings, July 1994, 15–22

[21] Sims, K. 1994. Evolving 3D Morphology and Behavior by Competition, Artificial Life IV Proceeding, ed. by R.Brooks & P.Maes, MIT Press, 1994, 28–39

[22] Sims, K. 1994. Karl Sims home page, http://www.genarts.com/karl/

[23] Skype Technologies S. A. 2002-. http://www.skype.com/

[24] Soda Creative Ltd. 2000-. Sodaplay. http://www.sodaplay.com/

[25] Taylor, T. J. 1999. From Artificial Evolution to Artificial Life, PhD thesis, Division of Informatics, University of Edinburgh.

[26] Tokui, N., and Iba, H. 1999. Empirical and Statistical Analysis of Genetic Programming with Linear Genome, in Proc. 1999 IEEE International Conference on Systems, Man and Cybernetics (SMC99), IEEE Press

[27] Tomita, K., Murata, S., Kurokawa, H. 2002. Graph Automata: Natural Expression of Self-reproduction, Physica D, 171, 4, 197–210

[28] Toth-Fejel, T. 2004. Modeling Kinematic Cellular Automata Final Report, NASA Institute for Advanced Concepts Phase I: CP-02-02

[29] Turing, A. M. 1952. The chemical basis of morphogenesis, Phil. Trans. Roy. Soc.

[30] Vreeken, J. 2002. A friendly introduction to reaction-diffusion systems. Institute for Information and Computing Sciences, Utrecht University.

[31] 反応拡散シミュレータ（Java アプレット）, http://www.bio.nagoya-u.ac.jp/~z3/research/rd_sim.htm

[32] 徳井 直生, 伊庭 斉志, 石塚 満. 1999. 汎用 LinearGP システムの構築と評価, 第 56 回情報処理学会全国大会予稿集 (2), 187–188