6J-04

効率的なコンピュータグラフィック開発のための論理型プログラミング言語TSUBAKIの開発と実装

中村 航† 才田聡子†

†北九州工業高等専門学校 生産デザイン工学専攻

1 はじめに

本研究では, 論理型言語 prolog の思想 [1] を取り入れ, 直感的にグラフィクスを開発できるプログラミング言語の開発を目指す.

本研究の最終的な目標は, 約 1 億個程度の恒星のリアルタイム描画が可能なプラネタリウムソフトの開発である. 恒星は UCAC4 という恒星カタログ [2] を利用し, さらに星間物質の分布を示す遠赤外線全天画像 [3] やダストトレイルのデータから求まる確率を基にした流星群 [4] の投影, 時間に応じて少しずつ動く天体, 空気の揺らぎによる星の瞬きなど, 本来の星空に近い空間を提供する.

当初, ゲームエンジンの Unity や OpenGL/GLSL を利用した開発を検討した. しかし, Unity は主に投影すること以外に機能を求めないプラネタリウムとしては不必要な機能も多く, OpenGL/GLSL もシェーダファイルの使い分けながら恒星や流星群などを表現することによる開発工程の冗長化が予想された.

そこで本研究ではプラネタリムさらには仮想世界シミュレータそのものを効率的に開発できるプログラミング言語を開発することに決め, TSUBAKI と名付けた.

C/C++系の手続き型言語の流れを組む言語は, 本質的に目標とする結果を得るまでの演算過程の説明に終始し, 構造化プログラミングやオブジェクト指向などの思想を取り入れ, 演算範囲を局所化するなどして可読性を維持してきた. 現状のコンピュータは計算機としてだけでなく, 娯楽やコミュニケーションなどの目的でも使われており, その目的が異なることにより生じる矛盾や複雑化への対処はプログラマに委ねられてきた. 例えば, 数値演算の結果ではない物理現象として存在する恒星や流星群をコンピュータ上で「見せかける」ことは, 各々の持つ本質的な「意味」との相違を生じ, いずれ矛盾に突き当たることが予想される.

そこで本研究による TSUBAKI の開発では, 一階述語論理を利用するなど, 論理型言語の思想を骨格としつつ, 「物理現象」の意味を持つ恒星や流星群と「数値演算」であるコンピュータの間に生ずる意味の相違に注目し, その意味の変換過程を記述, つまり物理現象を数値演算で近似することに注力することにした.

2 研究手法

2.1 言語的特徴

パラパラ漫画が 1 フレーム毎に何を表示すべきかを決定しているように, TSUBAKI は, ある一瞬の描画について記述し, ガードで処理すべき項目を決定する. また, 過去との連続性を保つために, 必要とされる過去のフレームの変数を保持し, 相対的な過去のフレームに簡単に参照することができる.

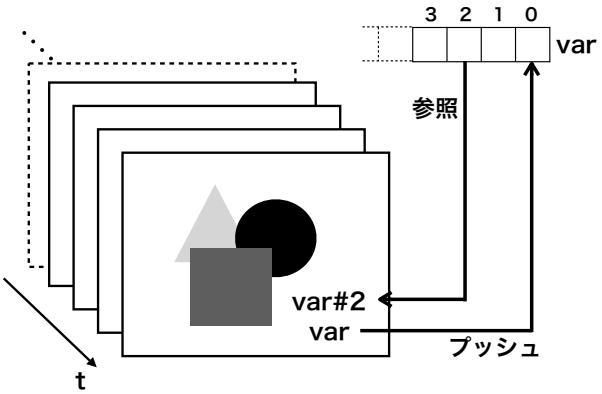

図 1: 過去データへの参照のイメージ

TSUBAKI が注目するのは, 意味 (メタ) の階層的つながりであり, 図 2 のように高レベルな意味を積み上げるようにしてプログラミングをしていく.

プログラミング言語が人間と計算機の対話手法の一つと考えれば, この図において最下層に位置するのは,

Development and Implementation of a Logic Programing Language TSUBAKI for an Efficient Design Environment of Visualization Products

Wataru NAKAMURA† and Satoko SAITA†

†National Institute of Technology, Kitakyushu College

802-0985 Kitakyushu Japan

{ad1731wn@apps.kct.ac.jp}

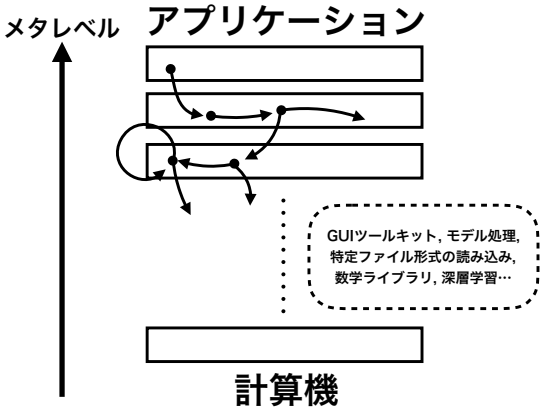

図 2: 意味の階層化のイメージ

計算機資源あるいはOSの提供する機能であり, 上層には計算機上で実現したいアプリケーションがある. 開発者は, この両者の関係性を積層するようにして組み上げていく.

2.2 文法

TSUBAKI コードの一例を以下に示す.

```
(namespace サンプル1)
(use サンプル2 サンプル3)

// ガード付き, 変数あり
((arg 変数宣言) (?(条件式) (定義) -: (説明)))
// ガードなし, 変数あり
((arg 変数宣言) (定義) -: (説明) )
// ガードなし・変数なし
((定義) -: (説明) )
```

図 3: 文法パターン

文法スタイルは Lisp に影響を受け, 括弧を多用する. ダブルスラッシュ"//"から右はコメント行で, 文法的に無視される. 一つのファイルに一つの名前空間があり, 空間名は必ず最初に定義し, 現在の空間内で使用する. 下層の空間は, use で指定する. TSUBAKI プログラマは, 定義を説明し, ガードで定義を選別し, 変数宣言でプログラムの汎用性を高める. 数値表現には, 10 進・8 進・16 進の整数値と実数値があり, 文字列もサポートする. また過去のフレームの変数に対しては, 以下のように (変数名#何フレーム前) の形で表現できる. #指定のある変数は, あらかじめそのフレーム数分メモリ領域を確保する.

((arg x) (... 省略... x#3 ... 省略...))

TSUBAKI は本質的に「コード生成」の言語であり, 実際の数値演算等に TSUBAKI の文法は関与しない.

2.3 実装方法

TSUBAKI はコード作成から実行まで以下の手順をたどる.

1. コンパイル部
   (a) 構文エラー解析
   (b) コードを固定バイト長のデータベースに集約
2. 実行部
   (a) 発火イベント地点よりデータベースを巡回し, 実行バイト命令列を生成
   (b) 命令列を VM が実行
   (c) 発火イベントが発生したら, (2a) から実行

本研究では最終的にプラネタリウムを含む仮想世界シミュレータの開発を目指すため, 命令列の実行段階では中間表現 SPIR-V を生成し、三次元グラフィック API の Vulkan を通してグラフィックを表示する.[5] マウスポインタの位置やウィンドウの情報など高度なグラフィックを実現する上で必要とされる諸機能は, 一フレームごとにそれらの値を定数として与えることを検討している. また, それらの情報はグラフィックツールキットの GLFW を利用することを検討している.

3 現状と今後の方針

現段階で文法エラーチェック機能は完成し, 構文エラー解析だけであればコマンドラインから確認することができる.

現段階では, 全て C/C++で実装する予定だが, Rust のように C 言語との相互運用性に優れながら, モダンな機能を提供する言語もあるため, 未着手の実行部については今後も検討の余地がある.

加えて, TSUBAKI の文法や思想についてもまだ荒削りな部分が目立ち, 今後もより洗練とした思想が完成するよう慎重に検討する.

参考文献

[1] 中村克彦: Prolog プログラミング, マイクロソフトウェア, 1984

[2] http://cdsarc.u-strasbg.fr/viz-bin/Cat?I/322A

[3] http://darts.jaxa.jp/astro/akari/

[4] R. H. McNAUGHT, D. J. ASHER, "Variation of Leonid maximum times with location of observer", MPS 34, 1999, pp. 975-978

[5] Graham Sellers: Vulkan Programming Guide, Addison Wesley, 2017