Puppy：ライブなコーディング入門環境

秋信 有花[1,a)]　坂根 万琴[1]　多田 拓[2]　倉光 君郎[1,b)]

概要：Puppy プロジェクトは，実用言語への橋渡しを目指した統合プログラミング入門環境を提供することである．物理エンジン上でプログラム実行環境が構築され，時系列にしたがった物体（オブジェクト）の動作としてプログラムの実行を体験しやすくなっている．加えて，ライブコーディングの導入や自然言語処理による支援などの新しい試みも目指している．本発表では，Puppy の試作システムを紹介し，特に中学校・高等学校レベルのプログラミング教育での活用を議論したい．

Puppy : Live Coding Environment for Introductory Programming

Yuka Akinobu[1,a)]　Makoto Sakane[1]　Taku Tada[2]　Kimio Kuramitsu[1,b)]

Abstract: Puppy is an integrated introductory coding environment that addresses a bridge to practical programming language comprehension. A program is executed on a physics engine with matters and time series, which makes it easy for novices to experience how a program runs. Besides, Puppy includes new attempts such as live programming and natural language processing supports. In this presentation, we will introduce an initial implementation of Puppy, in order to discuss a new approach to high school programming.

1. はじめに

（背景）近年，ネットワーク技術や人工知能の急激な発展により，我々の日常生活が大きく変化しつつある．このような現代社会の基礎素養として，「プログラミング教育」に関心が集まるのは自然な流れといえる．特に，米国「Hour of Code」が始まり，コンピュータ科学の教育の機運が全世界的に高まった．我が国でも，2020 年度からは小学校，その後，順次，中学校，高等学校の授業でプログラミング必修化が進む予定である．

（問題）プログラミング教育への関心が高まることにより，Scratch[2], [4] や Viscuit など優れた入門言語が開発されてきた．これらの入門言語は，小中学校で導入が始まり，プログラミング教育に活用されている．

これらの入門言語と C/C++ や Python などの実用プログラミング言語の間にギャップを感じることも少なくない．次のような理由が考えられる．

- **キーボード操作に慣れていない**：全角/半角入力の区別などは，キーボード操作に慣れていない初学者にとって難しい．
- **覚えなければならない文法が多い**：プログラムを書くには，まず文法を学習し，理解する必要がある．さらに，プログラムを自由に記述するためには，多くの文法を覚える必要が生じ，時間がかかってしまう．理解や記憶が曖昧のまま少しでも間違った記述をするとエラーが生じ，プログラムは動作しない．
- **コードを書くことで実現できることが子ども向けではない**：入門言語では，キャラクターを動かすことができるなど，子どもが楽しめる実行結果を簡単に得られるものが多い．しかし，実用プログラミング言語では学習時の子どもの感動が薄く，達成感を得難い．

プログラミング教育が小中学生から高校生へと高学年化が進んだとき，これらのギャップを踏まえた実用プログラ

[1] 日本女子大学理学部数物科学科
Department of Mathematical and Physical Sciences, Japan Women's University, 2-8-1 Mejirodai, Bunkyo-ku, Tokyo 112–8681, Japan
[2] 横浜国立大学大学院理工学府
Graduate School of Engineering, Yokohama National University, 79-1 Tokiwadai, Hodogaya-ku, Yokohama, Kanagawa 240–8501, Japan
[a)] m1616003ay@ug.jwu.ac.jp
[b)] kuramitsuk@fc.jwu.ac.jp

ミングへの入門となる環境が必要である．

（目的）Puppy プロジェクトは，実用言語への橋渡しとなる統合プログラミング環境を提供することである．

Puppy は，次のような方針で開発が始まった．

- **コード・ファースト** ：プログラミングの基礎は，構文を理解しコードを書くことである．「コードを書かせる」ことを避けるのではなく，自然言語処理などの支援を加え「コードを書く」ことを支援すべきである．
- **ライブ・コーディング** ：プログラムの動作は体感的に理解できるべきである．コードとともに，ステップや状態が変わる基本原理を理解しやすくすべきである．

このような方針に基づき，プログラミングの対象としてより具体的な物体（オブジェクト）を認識しやすくするため，Puppy は物理エンジン上に統合されたプログラミング演習環境として開発が進んでいる．

また，Puppy では，コードを書くことを支援する方法として，初学者でも書きやすい puppy 言語の提供や，自然言語処理によるコーディング支援を行なう．puppy 言語は Kid's Python として開発を進めている．文法をコンパクトにして覚える量を減らすことで，学習をスムーズに進められることが期待できる．さらに，子どものやる気を低下させる原因の一つであるエラーを減らすため，自然言語処理による支援を行なう．

本稿では，Puppy プロジェクトの紹介，現在の開発状況を報告する．2 節では，Puppy プロジェクトの動機について述べる．3 節では，Puppy の設計と開発状況について述べる．4 節で，本章を統括する．

なお，Puppy はオープンソースで開発を進め，8 月シンポジウムでは試作ソフトウェアのデモンストレーションを行なう予定である．

2. Puppy コンセプト

Puppy の特徴は，「時刻 t が変化すると，物体の状態が変わる」という物理モデルの上でプログラムを実行する点である．これは，プログラミングの理解を助ける役割を果たす．本節では，その動機を述べ，Puppy のコンセプトと実行モデルを概観する．

2.1 動機

プログラミング言語は，1960 年代の Fortran から数学知識を活用できるように設計されてきた．現在のプログラミング言語も，この設計思想は引き継がれ，変数や関数など数学に由来する記法を用いている．しかし，プログラミングの変数や関数は，数学のそれらとは大きく異なる．この違いは，次の代入式において端的に現れる．

$$x = x + 1$$

代入 (=) は，数学の等号と異なるものであるが，この背景をもう少し掘り下げると，次の漸化式で書き換えられる．

$$x_{t+1} = x_t + 1$$

ここで登場する t を理解するためには，少なくとも「ステップが進むと状態が変化する」という有限状態機械を感覚的であっても理解する必要がある．この実行モデルに対する理解が不十分であると，どんなに丁寧に文法を教えても，プログラムの実行を予測的に理解することができない．

2.2 物理エンジン

Puppy は，コンピュータのステップという概念を時刻 t に置き換え，「時刻 t が変化すると，物体の状態が変わる」という物理モデルから，プログラミングの理解を助けることを目指す．物理モデルを実現する手段として，物理エンジンを用いる．

物理エンジンとは，質量・速度・摩擦といった古典力学的な法則をシミュレーションするソフトウェアのことである．通常は，ミドルウェアライブラリとして，プログラミング言語から用いる．

物理エンジンを用いれば，物体は，物理法則に基づいて位置や形状が計算され，刻々と表示されるようになる．つまり，何のプログラム記述がなくとも落下するなどの動作が可能になり，少ない記述量で自然な動きが実現できる．

3. Puppy

本節では，現在開発中の Puppy の試作について報告する．

3.1 全体

Puppy は，物理エンジンと統合したプログラミング実行環境として開発している．図 1 は Puppy のスクリーンショットである．Puppy は，Web ブラウザ上でユーザインタフェースを提供する．利用者は，Web ブラウザを開いて画面右側のエディタでコードを書き，同じ画面でプログラムの実行を行なうことができる．実行結果は，画面左側のオブジェクトが描画される領域で見ることができる．また，ライブコーディングを導入し，その都度実行しなくても済むような環境を提供する．実用プログラミング環境で生じる「実行するためのコマンドを覚えられない」という初学者特有の問題 [5] を解決することを目指している．

プログラミング言語処理は，バックエンドの Web サーバ側で行う．利用者が書いたコードは，サーバ側でチェックされ，Web ブラウザ上で実行可能な JavaScript に変換される．変換されたコードは，Web ブラウザ上の Puppy 仮想マシン上にロードされ，実行される．

Puppy では，「コードを書く」ことをプログラミングの本質と考え，自由記述によるプログラムの実現を重視している．一方，初学者はまだキーボード操作に不慣れなことも多い．特に半角と全角の区別ができていない初学者は多

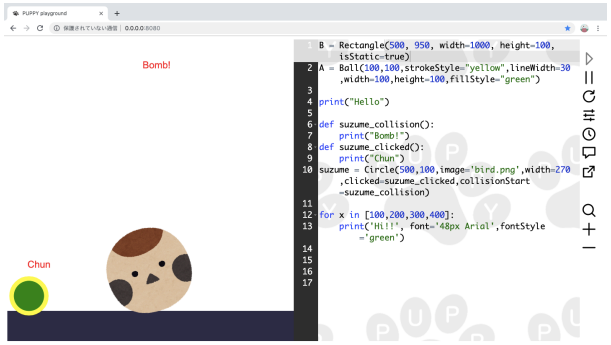

図 1　Puppy 統合環境

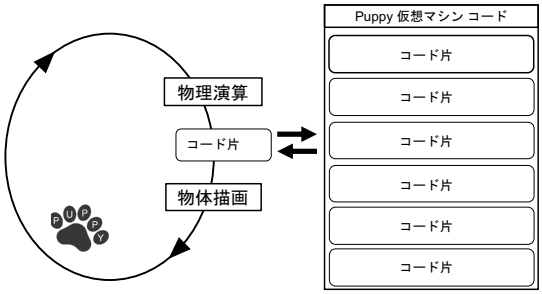

図 2　物理エンジンと Puppy VM

く，間違って全角を使ってしまったことによるエラーは起きやすい [6]．そのため，Puppy ではコードハイライトなどのコーディング支援が充実した ACE エディタ*1を導入し，全角入力時のフォローなどの日本語特有の支援も行なっている．

3.2　Puppy 仮想マシン

Puppy プログラムは，物理エンジン上の物体を記述し操作するものとする．Puppy では，物理エンジンの上にプログラム実行環境としての仮想マシンを構築する．物理エンジンは，時刻 t でループする周期実行の上で，物理演算と物体描画を繰り返している．この周期実行の中でブロッキングなしにプログラムを動かすため，図 2 のようにコード片に分割して，それらを順次，実行するようになっている．それぞれのコード片は末尾に yield を挿入することで分割している．yield の呼び出しごとに物理演算と物体描画を行なうことで，「時刻 t が変化すると，物体の状態が変わる」という物理モデルを体感できるようにしている．

Puppy 仮想マシンでは，軽量な物理エンジンである Matter.js*2を採用している．プログラミングでは，文字列による情報表示が特に重要になるため，Matter.js のレンダリング機能は独自の拡張を加え，変数の値を簡単に物体上に投影して表示可能になっている．単に文字列が表示されるだけの実用言語とは異なり，子どもが楽しくコードを書くことができるような実行状態を提供する．

*1　https://ace.c9.io/
*2　https://brm.io/matter-js/

```
B = Rectangle(500,950,width=1000,
    height=100,isStatic=True)
A = Ball(100,100,strokeStyle="yellow",
    lineWidth=30,width=100,height=100,
    fillStyle="green")

print("Hello")

def suzume_collision():
  print("Bomb!")
def suzume_clicked():
  print("Chun")
suzume = Circle(500,100,
         image='bird.png',
         width=270,
         clicked=suzume_clicked,
         collisionStart=suzume_collision)

for x in [100,200,300,400]:
  print('Hi!!',font='48px Arial',
    fontStyle='green')
```

図 3　puppy 言語による例

3.3　puppy 言語

Puppy システムでは，初学者でもコーディングしやすい puppy 言語を提供する．puppy 言語は Kid's Python として文法定義を進めている．文法定義をコンパクトにすることで，文法を覚える負担を軽減できると期待している．

図 3 は，puppy 言語のコード例である．puppy 言語の文法は Python のサブセットである．以下に，puppy 言語がサポートしている言語機能をあげる．

- **データ**：論理値，数値（整数と浮動小数点数の区別はない），文字列，リスト，物体
- **演算子**：上記の基本データの Python 演算子を原則サポート．
- **変数**：グローバル変数とローカル変数を区別する．
- **制御構造**：if 文と for 文のみサポートする．while 文は無限ループを避けるためにサポート外．
- **関数**：def 文による関数定義，再帰定義を認める．ラムダ式と第一級関数．
- **出力**：print 関数．
- **物体（オブジェクト）の出現**：Circle や Rectangle などを Play Ground 上に出現させることができる．オプション指定．画像を物体に貼り付けることも可能．

また，型推論による型検査を部分的にサポートし，型エラーをプログラマーに提示する．

3.4　自然言語処理によるコーディング支援

Puppy では，初学者の学習を支援する新しい試みとして，自然言語処理によるプログラミング支援を検討している．機械学習と自然言語処理の技法 [3] が現在大きく進展し，実行可能なコードに近似することも可能になっている．Puppy では，このような技術発展を取り込んで，エラーが生じることなく動くコードを生成し，学びやすさを向上さ

```
B = 壁(500,950,幅=1000,高さ=100)
A = ボール(100,100,幅=100,高さ=100,
    色は緑,よく跳ねる)
```

図 4 自然言語で記述されたコード例

```
B = Block(500,950,width=1000,
          height=100)
A = Ball(100,100,width=100,
         height=100,fillstyle='green',
         restitution=0.8)
```

図 5 図 4 の変換後のコード例

せる計画である．

puppy 言語のベースとなっている Python でよく生じるエラーとして，NameError があげられる [1]．NameError は，未定義な語が入力されたときに生じるエラーである．そこで，Puppy では，未定義な語が入力されたとき，定義済みの語に置き換えるコーディング支援を行なう．未定義な語が入力された場合，word2vec のモデルを用いて指定された未定義の語と辞書内で定義された語の類似度を計算し，類似度が最大の語を返す方針である．

図 4 は，自然言語を用いて記述したコード例である．「色は緑」や「よく跳ねる」などの自然言語で書かれたものをそのまま実行することはできない．図 5 のように，自然言語を実行可能なコードに置き換えることで，エラーが生じることなく動作するプログラムに変換される．

将来は，単語単位だけではなく，語句単位の自然言語を処理することを考えている．また，学習モデルを自作し，処理速度を向上していく予定である．

4. むすびに（今後の展望）

Puppy は，物理エンジンをベースにしたシミュレーション環境と統合されたプログラミング環境である．プログラムの対象は，実世界をモデル化した物体（オブジェクト）となり，コードと動作の対応がとりやすくなっている．本稿では，Puppy の開発状況を紹介した．今後は，Puppy 上で puppy 言語や自然言語処理によるコーディング支援など，新しい学習支援の試みの評価実験を進める予定である．

謝辞 Puppy は，著者の一人が座長を務めた「情報処理学会プログラミングシンポジウム (2018)」のパネル「高校生とプログラミング」の議論に触発されて開発が始まった．

参考文献

[1] Kohn, T.: The Error Behind The Message: Finding the Cause of Error Messages in Python, *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, New York, NY, USA, ACM, pp. 524–530 (online), DOI: 10.1145/3287324.3287381 (2019).

[2] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M. and Rusk, N.: Programming by Choice: Urban Youth Learning Programming with Scratch, *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, New York, NY, USA, ACM, pp. 367–371 (online), DOI: 10.1145/1352135.1352260 (2008).

[3] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J.: Distributed Representations of Words and Phrases and Their Compositionality, *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, USA, Curran Associates Inc., pp. 3111–3119 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2999792.2999959⟩ (2013).

[4] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y.: Scratch: Programming for All, *Commun. ACM*, Vol. 52, No. 11, pp. 60–67 (online), DOI: 10.1145/1592761.1592779 (2009).

[5] 岡本 雅子: ぺた語義：はじめてのプログラミングとつまずき, **情報処理**, Vol. 56, No. 6, pp. 580–583 (2015).

[6] 河村 一樹: 一般情報教育におけるプログラミング教育のあり方について, Technical Report 16, 東京国際大学商学部 (2011).