Puppy: プログラムの直感的理解を目指した新しいプログラミング入門環境

多田 拓[1]　秋信 有花[2]　坂根 万琴[2]　倉光 君郎[2,a)]

概要：プログラミング初学者がプログラムを理解するにはどうすればよいのか? 従来のプログラミング教材はソースコードに対する出力結果を提示するような，入出力からプログラムの理解を促す手法が一般的であった．しかし，入出力のみでプログラムを理解することは難しいため，教材や教員がプログラムの説明をすることがほとんどである．我々は，ブラックボックス化されてしまうプログラムの動作を可視化することでプログラムの理解を助けるプログラミング入門環境 Puppy を開発している．Puppy では，ハードウェア（実体のあるモノ）はソフトウェアより理解しやすい点に着目して，物理シミュレーションとプログラムの時間変化を対応づけることによるプログラム動作の可視化手法を実装している．本発表では，Puppy によるプログラム動作の可視化手法と Puppy の開発状況について報告する．

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）従来，プログラミングはコンピュータ科学や情報工学を専攻する学生，さらにサイエンスやエンジニアリングを学ぶ理工系の学生に対して，専門教育を行なってきた．しかし，プログラミング教育の裾野が広がることで，必ずしもモチベーションの高い学生だけが対象とならない．そこで，プログラム理解の本質的な困難さを解明し，ソフトウェア技術を活用し，プログラム理解を促進することが重要となる．

先駆的研究の例として，MIT メディアラボ の Scratch[1], [2] は，キーボード操作に不慣れな小学生に対し，ブロックによるビジュアルプログラミングを，兼宗らのドリトル [3] は，英語に不慣れな日本人学生向けに日本語プログラミングを提案し，対象にあわせた困難さの解消を図っている．これらは，プログラムの入力インターフェイスを学生向けに扱いやすくする取り組みであったが，本研究では，小学生より少し学年が進んだ高校生程度の学力がある学生を対象に，「プログラム理解の困難さ」に着目する．

従来のプログラミング教材はプログラムの入出力を提示し，そのプログラムの詳細な動作は教材や教員が説明するという方式がほとんどである．そのため，学生がプログラムを理解するためには，質の高い説明を必要とし，誤解を招くような表現に配慮する必要があった．我々は，ハードウェア (実体のあるモノ) はソフトウェアより理解しやすい点に着目し，プログラム理解をより直感的にする方法を提案する．

（目的）Puppy は，物理世界の直感理解を可能にする統合プログラミング入門環境である．次のような特徴を持つ．

- **物理シミュレーション** – プログラムは，物理シミュレーション上で物理法則に影響を受けながら動作する．
- **簡易 Python 言語** – 本格的なプログラミングにステップアップしやすいように，プログラミング教育で定評のある Python の簡易版を用いる
- **統合 Web 開発環境** – 教員や学生がインストールや設定作業なしに Web ブラウザひとつで利用可能なプログラム実行環境とする

Puppy プロジェクトの最終的な目標は，高校教員と高校生に対して，プログラミングを教えやすく学びやすいソフ

[1] 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
[2] 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
[a)] kuramitsuk@fc.jwu.ac.jp

トウェアを提供することである．本研究で開発した Puppy は，オープンソースとして研究成果を公開するするだけでなく，以下のサイトから利用可能な状態になっている．

https://playpuppy.github.io/

本稿の残りの構成は以下の通りである．2 節では，Puppy 開発となった背景と動機，さらに Puppy の裏側にあるアイディアについて述べる．3 節では，Puppy Playground! を紹介する．4 節では，Puppy Playground!によるプログラム動作の可視化について述べる．5 節では，関連研究をまとめる．6 節では，本論文を総括する．

2. プログラムの理解はなぜ難しいのか？

本節では，Puppy 開発に至った背景と動機，その背後にあるアイディア（着想）について述べる．

2.1 経緯

著者の一人は，情報処理学会「プログラミングシンポジウム 2018 (1 月)」で，「高校生向けのプログラミング」というパネルの座長を担当した．プログラミング必修を前に，100 人を超える大学教員がパネルに参加し，各自の高大連携プログラムの経験を交え，白熱した議論となった．その議論の中，以下の共通認識を得ることができた．

- ゲームのような題材，ロボット制御などの題材は扱いやすい
- アルゴリズムなど抽象的な題材は理解されにくい
- そもそも有限状態機械モデルが難しい

このような議論を通して，ロボット制御などのフィジカル・プログラミングは，一見するとオモチャで遊んでいるように見えるが，「時系列にしたがって変化する物理感覚」から理解を高めるという点で，プログラミング教育としてのある程度の妥当性があるが確認できた．

2.2 フィジカル・プログラミング

現在，フィジカル・プログラミング [4] は，ロボット制御などを題材として人気の教材になっている．実際にロボットが動くというだけでなく，プログラムによる動作が予想しやすく，理解しやすいためといえる．一方，フィジカル・プログラミングを導入するには，スペースや設備の面で負担が少なくない．また，アクチュエータ（モーター）が期待どおりに動作しない，センサーの読み込んだ値が環境に依存するなど，組込みプログラミング特有の難しさがある．また，フィジカル・プログラミングは，在宅学習などの演習量が確保できず，プログラムは理解しやすいが必ずしもプログラミング力の向上に望ましくないという研究 [5] もある．

Puppy は，フィジカル・プログラミングを仮想的に実現することで，直感的な理解と PC ひとつによる演習環境を実現することをゴールの 1 つとしている．

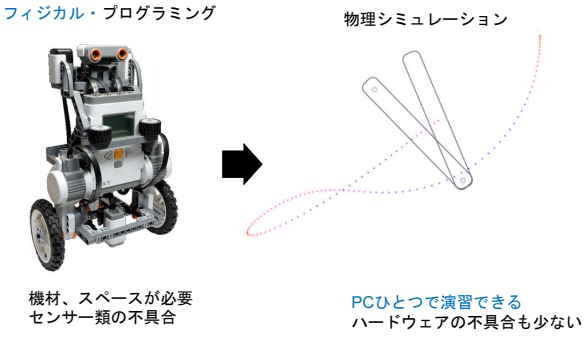

図 1　仮想的なフィジカル・プログラミングの実現

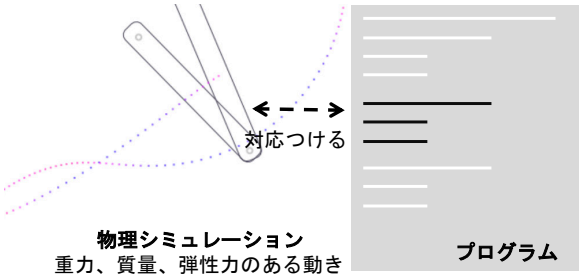

図 2　Puppy コンセプト

2.3 プログラム動作の可視化

最後に，Puppy によるプログラム動作の可視化について述べる．

図 2 は，プログラム動作の可視化のアイディアを図示したものである．プログラム動作は実体を体感できないが，物理シミュレーションと対応づけることによってプログラム動作を可視化する．例えば，円を繰り返し出現させるようなプログラムを実行したときには，円が**段階的**に出現する様子をシミュレーションする．この**段階的**というのはデバッガのステップ実行のようなものであり，プログラムをあえてゆっくりと実行することで人間がプログラムの動作を視覚的に追えるようにする．

プログラムの動作を可視化し理解させることができれば，自身でバグを発見したり，さらに発展的なプログラムへの修正などの自発的な行動を促すことができると期待している．

3. Puppy Playground!

Puppy Playground!は，プログラミング入門教育向けの統合演習環境である．物理シミュレーションと統合されたビジュアル環境をベースにしており，プログラム動作を直感的に理解しやすくなるように設計されている．

3.1 システム概要

Puppy Playground!は，Web ブラウザでユーザインタフェースを提供する．利用者は，Web ブラウザを開いてコードを書き，同じ画面でプログラムの実行を行うことが

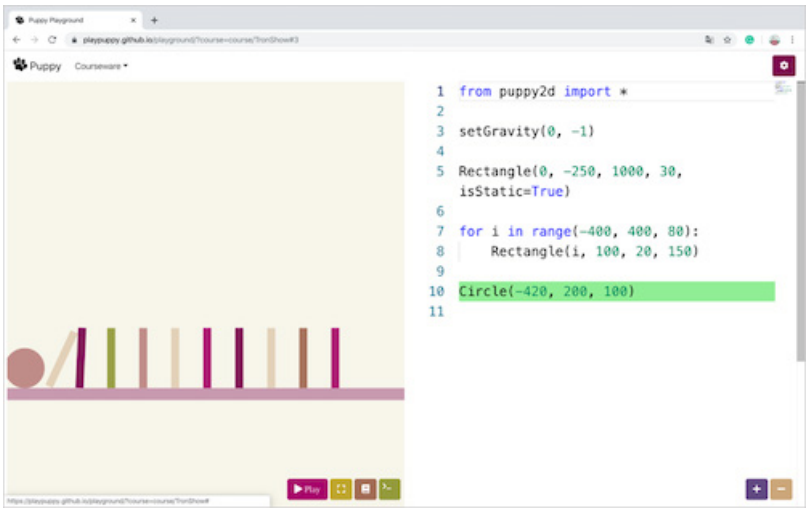

図 3 Puppy Playground!

できる．図 3 は，Puppy Playground!を示している．右側で編集したコードは，左側で実行し動作を確認することができる．

図 4 は，Puppy Playground!のシステム構成図を示している．Puppy Playground!は，Web ブラウザを用いるが，サーバーレスなアーキテクチャになっており，一旦ページを開くとオフラインでも利用することができる．エディタで編集されたソースコードは，トランスパイラ技法により，物理エンジンライブラリ Matter.js を用いた JavaScript に変換される．

3.2 ユーザインターフェース

Puppy Playground!のユーザインターフェイスには 2 つの主要な要素がある．一つは Puppy 言語を記述するソースコードエディタである．シンタックスハイライトや入力候補の表示などの一般的な統合開発環境で提供されている機能や，全角文字のハイライトやコンパイルエラーを背景色の変化でフィードバックするなどのするなどの入門者をサポートする機能を実装している．また，プログラムの実行と同期して，ソースコードの対応している行がハイライトされる．ソースコードエディタは Monaco Editor ライブラリを使用しており，API を利用してコーディング支援機能を実現している．

もう一つの要素はプログラムの実行結果を視覚的に表示する領域である．HTML5 から導入された 2D グラフィックスを描くことができる canvas タグによって実装されている．Puppy Playground!は，軽量な物理エンジンである Matter.js[*1]を採用している．ソースコードエディタに記述されたソースコードはトランスコンパイラによっての Matter.js API を呼ぶ Javascript へと変換され，実行される．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．print() 文などの表示も Matter.js のレンダリング機能で行なっている．

物理エンジンの動作中は 60[fps] の速度で描画を繰り返し，物理シミュレーションを表示する．

3.3 トランスコンパイラ

Puppy 言語処理系は，Puppy ソースコードから JavaScript ソースコードへの変換器，トランスコンパイラとして開発されている．

Puppy のシンタックスは，TPEG [6] で与えられる．TPEG は，解析表現文法の一種であり，宣言的な構文木構築やシンボルテーブルによる状態管理の拡張が追加されている．これらの機能により，Python スタイルのインデント依存のブロックの構文解析が可能となっている．

また，Puppy トランスパイラは，型推論による静的な型検査をサポートし，簡単な型エラーを検出することができる．

3.4 物理エンジン

Puppy Playground!は，抽象的なプログラム動作を物理現象にマッピングすることで，プログラミング理解を助ける．これは言い換えると，「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．

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．

[*1] https://brm.io/matter-js/

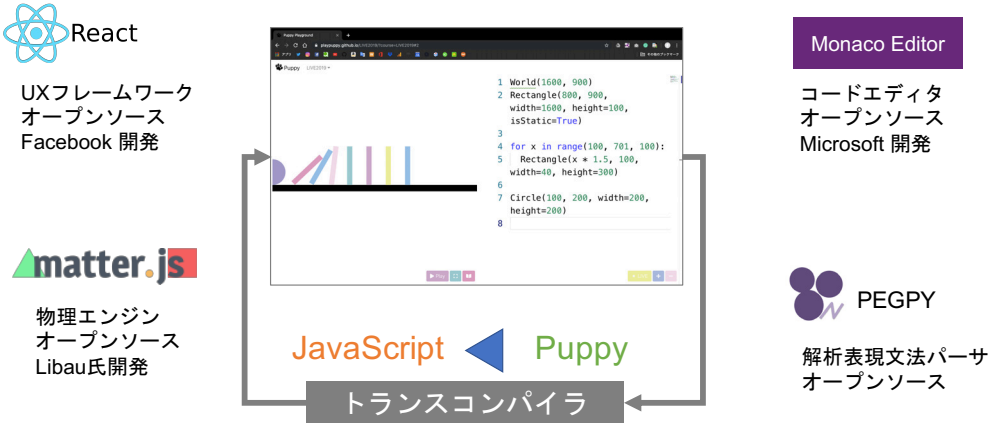

図 4　Puppy Playground!システム構成図

4. Puppy Playground!によるプログラム動作の可視化

Puppy Playground!では Puppy 言語が Matter.js を用いた JavaScript に変換されることによって Puppy プログラムによる物理シミュレーションを実現している．本節では，Puppy 言語の仕様の一部を紹介するとともに，物理シミュレーションとの連携を用いたプログラム動作の可視化手法について述べる．

4.1 Puppy 言語

Puppy 言語は，人工知能開発の標準言語となった Python 言語の簡易版（サブセット）として設計する．これにより，Puppy 言語を学ぶことで大学進学後のより専門性の高いプログラミング教育にスムーズに連携できる．一方，Python の簡易版であるが，高校生向けの情報オリンピックの問題を回答できる記述力は持つようにする．これにより，アルゴリズムなどのコンピュータ科学の世界に関心が広がるようにしたい．

4.1.1 言語設計

Puppy 言語は，限られた学習時間内でコーディングが可能なように，Python の文法やデータ構造，利用可能なライブラリを制限し，学習負担を軽くしている．

- 制御構造は，`if` 文と `for` 文のみである．無限ループを避けるため，`while` 文をサポートしない．
- データ構造は，論理値，数値，文字列とオブジェクト（物体），それらのリストである．
- 関数定義は，再帰関数やラムダ式を認める

詳細な言語仕様については付録を参照していただきたい．

4.1.2 物体操作によるオブジェクトプログラミング

Puppy 言語は，物理エンジンと統合されたビジュアルプログラミングである．matter.js のライブラリを通して，物理エンジン上の物体を直接操作することができる．物体オブジェクトはコンストラクタで生成し，それと同時に (x,y) 座標を与えて配置を行う．

Circle(x,y, **options)　　円形の物体を生成する
Rectangle(x,y, **options)　長方形の物体を生成する

物体の物理的な特性は，**options において与える．次の例は，ボールの幅 width=50, 色 color=”red”のボールを生成し，座標 (500,500) に置くことを意味する．

```
Ball = Circle(500, 500, width=50,
  color="red")
```

オプションは，オブジェクト生成後，フィールドアクセスを通して変更することができる．

```
Ball.fillStyle="blue"
Ball.setPosition(x, y)
```

このような物体操作を通してオブジェクト指向プログラミングの基礎（メソッド呼び出し，プロパティ）を学ぶことができるようになっている．

4.2 プログラム動作の可視化

Puppy ではプログラムの実行と物理エンジンが統合されている．この統合を実現するため，図 5 に示すとおり，Puppy ではプログラムをいくつかのコード片に分割している．これらのコード片を時分割された物理エンジンのイベントループ内で実行することによってプログラムと物理シミュレーションの同期を実現している．

さらに，Puppy Playground!ではプログラムの実行中の行がハイライトされるため，実行された行とその実行結果を視覚的に捉えることが可能である．

5. 関連研究

プログラミング教育の重要性が世界的に高まっている中，子どもたちがプログラミングをスムーズに学ぶことができるように，様々なタイプの教育用プログラミング言語の開発が盛んに行われている．

Scratch[1], [2] は，MIT メディアラボが開発した，ビジュアルプログラミングアプリケーションである．特に 8 歳から 16 歳向けにデザインされており，Web 上で UI が提供

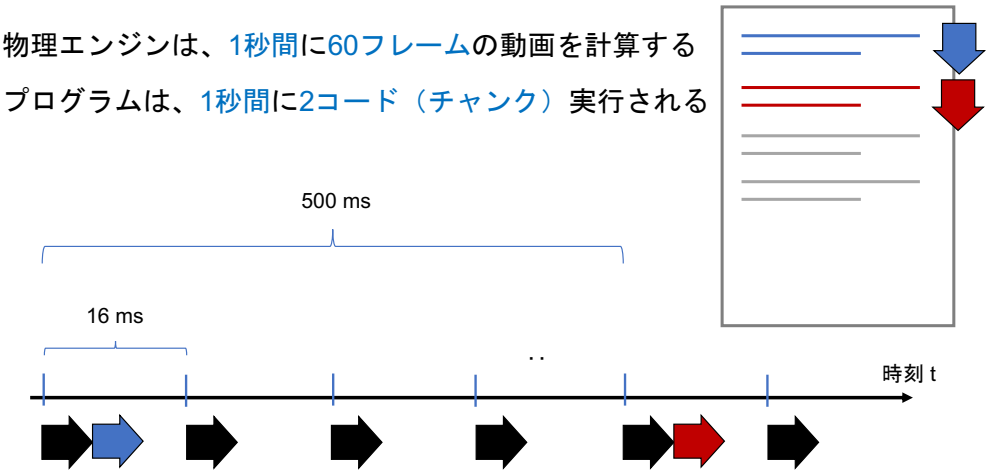

図 5 物理エンジンとプログラム実行
右側で編集したコードは，左側で実行し動作を確認することができる．

されている．Scratch の特徴は，ブロックを組み立てることでプログラムを記述するブロックベースの言語という点である．マウス操作によって直感的にプログラムが記述できるため，キーボード操作に慣れていない子どもたちでも簡単にプログラミングを体験することができる．

ドリトル [3] は，教育用に設計された日本語プログラミング言語である．日本の中学校・高等学校の教科書等に採用されており，対象は小学校から大学までとされている．ドリトルの特徴は，日本語によるプログラミングを実現している点である．まだアルファベットや英語に慣れていない子どもたちがプログラムを記述しやすいよう配慮されている．

Racket[7], [8] は，関数型プログラミング言語を提供することで，初学者の困難さを解消する狙いである．算数や数学を学んできた中高生にとって，プログラミング特有の $x = x + 1$ のような代入式の概念を理解することのは難しい．関数型プログラミング言語を用いることで，代入式の概念を無くし，理解が難しいポイントを減らしている．

また，LEGO Mindstorm[9], [10] や Micro.bit [11] などのハードウェアを提供することで，プログラミング学習をスムーズに進めようとする取り組みもある．現実にある物理的な物体をプログラムで操ることで，子どもたちのプログラミングに対する興味・関心を引き出すことができる．さらに，物体の動きを通して実行結果を確認することができるので，直感的にプログラムを理解する手助けにもなる．

6. まとめと今後の展望

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，コードとプログラム動作の対応が直感的にわかりやすくなることを目指している．本稿では，Puppy のコンセプトと開発状況を紹介した．

Puppy の開発は始まったばかりである．Puppy 言語のプログラミング学習に適した言語仕様化，ライブプログラミング等を導入した Puppy Playground!機能の拡張，自然言語処理によるコーディング支援など，プログラミング教育を研究するプラットホームとして活用する予定である．また，高等学校の教員がすぐに利用しやすいコースウェアの開発を進め，多くの若者がプログラミングを学びやすい環境を整えてゆきたい．

謝辞 Puppy 開発を進めるに辺り，有意義なユーザ評価をいただいた日本女子大学理学部サマースクール 受講生，神奈川県立厚木高等学校大学訪問授業の受講生に感謝する．

参考文献

[1] 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).

[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] 兼宗　進，御手洗理英，中谷多哉子，福井眞吾，久野靖：学校教育用オブジェクト指向言語「ドリトル」の設計と実装，情報処理学会論文誌プログラミング（PRO），Vol. 42, No. SIG11(PRO12), pp. 78–90 (2001).

[4] Montemayor, J., Druin, A., Farber, A., Simms, S., Churaman, W. and D'Amour, A.: Physical Programming: Designing Tools for Children to Create Physical Interactive Environments, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, New York, NY, USA, ACM, pp. 299–306 (online), DOI: 10.1145/503376.503430 (2002).

[5] Fagin, B. S. and Merkle, L.: Quantitative Analysis of the Effects of Robots on Introductory Computer Science Education, *J. Educ. Resour. Comput.*, Vol. 2, No. 4 (on-

line), DOI: 10.1145/949257.949259 (2002).
[6] Kuramitsu, K.: Nez: Practical Open Grammar Language, *Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2016, New York, NY, USA, ACM, pp. 29–42 (online), DOI: 10.1145/2986012.2986019 (2016).
[7] Ramsey, N.: On Teaching *How to Design Programs*: Observations from a Newcomer, *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ICFP '14, New York, NY, USA, ACM, pp. 153–166 (online), DOI: 10.1145/2628136.2628137 (2014).
[8] Yoo, D. and Krishnamurthi, S.: Whalesong: Running Racket in the Browser, *Proceedings of the 9th Symposium on Dynamic Languages*, DLS '13, New York, NY, USA, ACM, pp. 97–108 (online), DOI: 10.1145/2508168.2508172 (2013).
[9] Lawhead, P. B., Duncan, M. E., Bland, C. G., Goldweber, M., Schep, M., Barnes, D. J. and Hollingsworth, R. G.: A Road Map for Teaching Introductory Programming Using LEGO&Copy; Mindstorms Robots, *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '02, New York, NY, USA, ACM, pp. 191–201 (online), DOI: 10.1145/782941.783002 (2002).
[10] 志田駿介，井出真広，倉光君郎：Mindstorms NXT を対象とした Konoha 処理系のコンパクト化，情報処理学会論文誌プログラミング（PRO），Vol. 6, No. 4, pp. 1–9 (2013).
[11] Siever, B. and Rogers, M. P.: Micro: Bit Magic: Engaging K-12, CS1/2, and Non-majors with IoT & Embedded, *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, New York, NY, USA, ACM, pp. 1237–1238 (online), DOI: 10.1145/3287324.3287527 (2019).