ChatGPT を利用したプログラムの翻訳による プログラミング学習手法の提案

中村隆之[†1]　長谷川忍[†2]

概要：プログラミング言語の多くは英語を主体としている．最近利用されているプログラミング言語，例えば C#はコーディングルールにクラス名，変数名あるいはメソッド名をフルスペルで書く事が記載されており，英語ネイティブの学生にとっては可読性を増す効果がある．一方で，日本をはじめとする非英語圏の学生にとっては英語の理解度が原因となりフルスペルでの記載によりソースコードの意味を掴みにくくなっている可能性がある．日本語は特に英語と文法上の違いが大きく，プログラミング時に負担を感じる一因となっていると推測される．本研究では，ChatGPT を利用しソースコード中の英語で記述されている変数名，メソッド名等を日本語に翻訳し，翻訳前のコードと共に提示するコード翻訳システムを提案する．本コード翻訳システムを，日本語を母国語とする学生被験者 20 名に評価させた結果，より複雑な内容ではコード理解度が増す事が判明した．主観評価ではこのシステムを良いと答えた群と翻訳しないシステムを良いと答えた群が分かれ，翻訳ありのシステムを良いと答えた群は主観での英語理解度とプログラム理解度が低い傾向の群である事もわかった．

1. はじめに

本研究は，Large Language Model (LLM)の一つである ChatGPT を利用し，ソースコードの英語部分を日本語に翻訳する事で日本語母語話者がソースコードを理解しやすくする手法に関する研究である．

現在様々なコンピュータソフトウェアを開発するためのプログラミング言語が利用されている．それらのプログラミング言語はそれぞれの時代のニーズにあわせ，また目的に応じて再設計され，別のプログラミング言語と分化し進歩している．

プログラミング言語と同時にコンピュータの性能そのものも日々向上しており，コーディングのルールもそれに合わせて変化している．FORTRAN，COBOL，C 言語といった言語が主流だった時代には変数名や関数名といった名前は，短い変数名や略された関数名を利用する事が一般的であった．しかし Visual Studio 等のコードエディタの補完機能等が一般的になった現在は長い変数名であってもミスなく入力できるため，変数名，関数名，あるいはクラス名は可読性のために略されずフルスペルで定義される事が一般的となっている．C#の場合は .NET Framework での開発行うためのコーディングルールが定められ，その中でも基本的に変数名やメソッド名のフルスペルで記述する必要がある事が記載されている[1]．フルスペルで記載されていれば省略される場合よりもコードの意味は理解しやすくなり可読性が上がると言える．

このような流れにより，英語が母国語の，あるいは英語で教育を受けた学習者にとってはコードの可読性は以前に比較して増している事が考えられる．しかし，英語と異なる言語，例えば日本語を母国語とし英語に苦手意識を持つ学習者にとっては，英語のフルスペルで記述されたコードの場合むしろ可読性が落ちている可能性もある．Guo ら[2]は，英語を母国語としてない人々はコードの読み書きや英語とプログラミングの同時学習という障壁に直面していることを調査で示した．また，Guzman ら[3]は英語を母国語としない留学生のプログラミング学習者が言語の問題により授業外の学習に時間をかける傾向があることを報告している．

特に日本語においては，英語とは語順の違いや表意文字である漢字と表音文字であるひらがなおよびカタカナが混在する等言語的な違いが大きいと考えられる．最近のビデオゲーム開発のように，ゲームエンジンが提供する機能が複雑になり，ゲームエンジンが提供する API，クラス名や関数名(メソッド名)が長い英語で記述されるケースもあり，さらにメソッド名は多くの場合動詞＋目的語のように日本語とは語順も異なる．そのようなケースでは英語が理解できない事でプログラムの内容を理解できない傾向が特に強い可能性がある．

本研究の目的は，日本語を母国語とするプログラング学習初心者に対して，他人のソースコードを理解しやすくする目的のシステムを開発する事である．具体的には，ソースコードの構造をそのままに ChatGPT を利用して日本語に翻訳する．次に日本語に翻訳されたコードと同時に元のソースコードを同時に閲覧できるようにする．開発したシステムにより他人が書いた英語のソースコードの内容を日本語のプログラミング初学者にとって短時間で理解できるようにする事が期待される．本研究ではそのシステムの有用性と同時にデメリット等を確認する事が目的となる．

2. 研究手法

本研究では，まずコード翻訳閲覧システムの開発を行う．次にそのシステムについて日本語を母語とするプログラミング学習者への有用性を評価する．最後にシステムの有用性について分析する．

†1 東京工芸大学/北陸先端科学技術大学院大学
†2 北陸先端科学技術大学院大学

2.1 コード翻訳システムの開発

対象とするコードはコメントおよび全ての変数, あるいは関数，クラス名が英語により記述されているコードを対象とする．本研究では条件に一致する言語処理系として，ゲームエンジン Unity 上で利用される C# 言語を用いる．

コード翻訳システムの画面は図 1 のようなイメージである．ウインドウが左右に分かれており，左にはオリジナルのコード，右には対応する翻訳されたコードが表示されている．左右のウインドウの縦方向スクロールは連動しており, 右ウインドウでは同一行の対応する翻訳が確認できる．また，モダンなコードエディタでは一般的なシンタックスハイライトは左右画面とも有効となっている．シンタックスハイライトとは，変数，予約語等を色分けやフォント属性変更する事でコードをわかりやすくする機能である．

図 1 提案システムのイメージ(コード B の一部)

実験用コード翻訳閲覧システムではコードの編集はできないものとする．これは本コード翻訳システムでは，一つのソースコードファイルを翻訳するのに数十秒から数分程度を要するためである．本実験ではコード翻訳は，静的に行われる．

本実験ではコードは HTML に変換される．変換時にシンタックスハイライトを行うライブラリとして highlight.js [4]を利用している．翻訳されたコードの一部は自動でハイライトが行われないため，手動でシンタックスハイライトを行っている．

コード翻訳には LLM として代表的な Open AI 社の ChatGPT (GPT 4)を利用する．ChatGPT は自然言語のテキストタスクだけではなくプログラムコードの処理等も可能な事が知られている．

図 2 が実際にコード翻訳実験に利用した ChatGPT 向けのプロンプトである．本研究においては，コードの予約語にあたる部分はそのままに, 変数名, 関数名(メソッド名)を日本語に変換するプロンプト設計方針としている．これは，英語理解度が低いケースでコードの理解が妨げられる一因がフルスペルで書かれた変数名やメソッド名なのではないかと考えられたからである．

実験時にコメントがあるとコード翻訳の評価に影響があるため本実験ではコメントは出力しないようにしている．

```
EtoJCodeという処理について以下に定義します。

Unity C#スクリプトの英語部分を日本語に変換してください。変換されたコードは正しく動作しなくても問題がありません。ただし下記のルールに従ってください。

1．行数は変更しないでください。コメント等の実行時に意味がない部分も行数を変更しないようにお願いします。
2．英語から日本語に変換するのは変数名、関数名(メソッド名)、構造体のメンバ名です。UnityのAPIとして準備されているClassはクラス名以外のメソッド名も英語から日本語に変換してください。
3．構造体のメンバー名も英語から日本語に変換してください。
4．変換された出力はコードのみで、それ以外の出力は必要ありません。
5．コメントは出力しないようにお願いします。
わかったらOKを出力
```

図 2 コード翻訳用の ChatGPT プロンプト

2.2 コード翻訳システムの評価実験

本実験ではゲームエンジンである Unity 上の C#スクリプトを評価実験用の言語として採用した．そのため評価実験被験者募集の際に「ゲームエンジン基礎」の単位取得者である事を条件にスクリーニングを行なった.「ゲームエンジン基礎」は実験者が所属する東京工芸大学ゲーム学科内で，1 年時後期に実施される Unity 等のゲームエンジンについて基礎を学ぶ授業である．被験者は 20 名とした．最後まで参加した被験者には 500 円相当の電子ギフトを謝礼として送付した．

被験者は 2 つのコード(コード A およびコード B)についてコードの内容を理解しているかどうかを問う設問(問題 A および問題 B)に回答するテストを行い，そのコードの内容についての正答率と，回答開始までに要した時間を測定した．被験者は 2 つのコードについて内容に関する記述テストを行った．2 つのコードの順序と，本システムによる翻訳の有無を先に行うか後に行うかの 2 通りで，被験者を合計 4 つのグループにわけて実験を行った．4 つの被験者グループの条件を表 1 に示す．これにより，コード A の評価とコード B の評価を行う順序，および翻訳ありコードと翻訳なしコードの順序による効果を打ち消すことを意図している．

表 1 被験者グループ(4 グループ)条件

	「翻訳あり」が先	「翻訳あり」が後
問題 A が先	グループ 1	グループ 2
問題 B が先	グループ 3	グループ 4

2.2.1 評価実験に利用するコードについて

実験に使われるコードは, ゲームの実装に利用される基本的なコード 2 つ(コード A,コード B)を準備した．両スクリプトとも Unity 上の C# スクリプトである．

コード A は一般的かつ基本的なコードでプログラムの基本を理解している学生であれば理解できるレベルのコードである．一方コード B はコード A に比較するとやや複雑で，コードのライン数もコード A に比較して多い．

以下はコード A の解説である．プレイヤーを前後左右に動かし，マウスでカメラを回転させる．マウスのボタンでプレイヤーがジャンプする．

以下はコード B の解説である．コードは敵キャラクターの簡易的な AI となっている．プレイヤーが一定範囲に近づくと，敵キャラクターを追いかける．プレイヤーが一定範囲から離れると，初期の位置に戻る行動をする．演出として，敵キャラクターがプレイヤーを見つけた時は警告を表すマテリアルに変更され，元の位置に戻ろうとする時はマテリアルも元のマテリアルに変更される．

2.2.2 アンケートによる評価

被験者は実験室にてアンケートに回答した. アンケート時には 2 画面のモニタの内上部画面にコードを示した．その提示されたコードがどのように動作するかを被験者に下部画面のアンケートフォームに記述回答させた．記述回答する時間には問題 A, 問題 B それぞれに 10 分間の制限時間を設けた．

事前に 2 つのコードのそれぞれについて記述内容の採点基準を設定した．同時に記述回答を開始するまでの時間を測定した．記述内容の採点と時間それぞれを，本コード翻訳システムを利用した場合とそうでない場合で比較した．それにより，本システムの効果を検証する．問題 A(コード A に対応), 問題 B(コード B に対応)それぞれの満点は 100 点となるように配点を行った．100 点の配点の内訳は以下である．コード全体が何の処理を行っているかの概要についての配点が 20 点，コード全体の処理についての項目毎に 30 点，コードのうち指定された一つの関数内の処理内容の記述が 40 点，それらの処理順が合っていれば追加で 10 点としている．

また, コードの理解度を測るテスト問題の終了後に以下のアンケート項目を追加した. 1) 主観による英語の理解度に関する設問(5 段階)，2) 主観によるプログラミングの理解度に関する設問(5 段階)，3) 翻訳ありと翻訳なしのシステムどちらが良いか(7 段階)，4) 翻訳ありシステムについての自由記述，の 4 項目である．

3. 実験結果

この章では評価実験の結果を述べる．

3.1 翻訳システムありの場合となしの場合での得点比較

翻訳システムありの場合となしの場合において，被験者のコードの理解度を示す得点を比較した．比較は問題 A と問題 B に分割して実施した．正規分布を仮定できないため検定にはマンホイットニーU 検定を使用した．結果を表 2 に示す.

表 2「翻訳あり」「翻訳なし」それぞれの得点比較

	翻訳あり(中央値)	翻訳なし(中央値)	p 値
問題 A 得点	80	72.5	0.54
問題 B 得点	80	27.5	0.04

問題 A(プレイヤーキャラクターを操作するための一般的なコード)の場合は, 翻訳システムありの場合となしの場合での得点には統計的に有意な差は認められなかった．しかし問題 B(敵キャラクターの簡易的な AI)の場合は，翻訳システムありの場合となしの場合で統計的に有意な差が認められた($p < 0.05$)．問題 B の場合，翻訳ありの得点中央値は 80 点に対し，翻訳なしの得点中央値は 27.5 である．これは問題 B に関しては，翻訳システムありの場合について翻訳なしの場合に比較し有意に得点が高い事を示している．

3.2 翻訳システムありの場合となしの場合での回答開始までにかかった時間の比較

翻訳システムありの場合となしの場合で，各被験者が回答に取り掛かるまでの時間を測定し，比較した．比較は問題 A と問題 B に分けて行なった. 正規分布を仮定できないため検定にはマンホイットニーU 検定を使用した．結果を表 3 に示す.

表 3 「翻訳あり」「翻訳なし」それぞれの回答開始時間

	翻訳あり(中央値)	翻訳なし(中央値)	p 値
問題 A 回答開始(秒)	64	64	0.68
問題 B 回答開始(秒)	49	76	0.79

回答開始時間については, 問題 A および問題 B どちらも，翻訳ありの場合となしの場合で統計的な有意差を認められなかった．

3.3 主観の英語理解度とプログラム理解度の相関

プログラムに利用される英語の主観理解度と, プログラムの主観理解度について，スピアマンの順位相関分析を行った．その結果，英語の主観理解度とプログラムの主観理解度について有意な正の相関が見られた($\rho = 0.62$, $p < 0.01$).

3.4 「翻訳あり」「翻訳なし」のどちらが良いかの回答

翻訳ありが良いか無しが良いかについて被験者に 7 スケールで回答してもらった結果，頻度は図 3 のようになった．

主観の英語理解度とプログラム理解度をそれぞれ縦軸, 横軸に取り，「翻訳あり」が良いと答えた群(オレンジ)と「翻訳なし」が良いと答えた群(ブルー)をバブルチャートとしてマッピングした図になる(図 4).

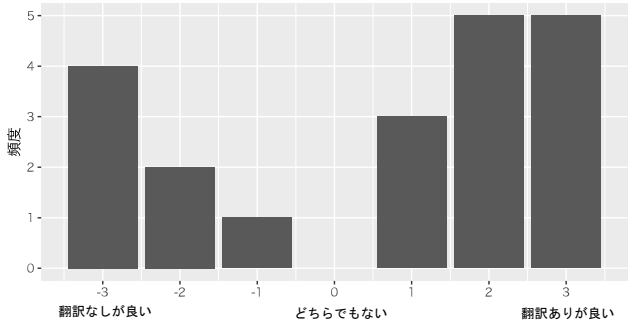

図 3 「翻訳あり」「翻訳なし」どちらが良いかの評価

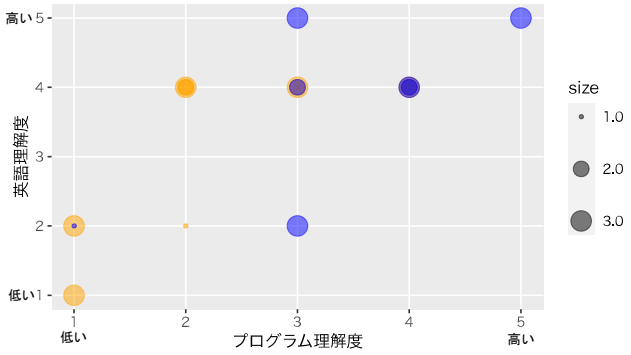

図 4 翻訳ありが良い(オレンジ)と翻訳なしが良い(ブルー)の分布

翻訳ありが良いと答えた群は主観での英語理解度とプログラム理解度が低い傾向にあることがわかる.

4. 考察

ここでは実験の結果から得られた事項について述べる.

4.1 翻訳システムによるコード理解度の変化

翻訳システムありの場合となしの場合で統計的有意差が認められたのは，問題 B であった．問題 B の回答にあたっては今回の実験条件では「翻訳あり」のほうがプログラムの理解度が高くなった事が推測される．しかし問題 A では統計的有意差は認められなかった.

問題 A に比較すると，問題 B はコードが長く，やや複雑である．また問題 A は Unity のゲームプログラミングにおいてはプレイヤーキャラクターを操作する基本的なコードである．本実験の被験者が, Unity のプログラミングの基本を教える授業の単位取得者である事を条件にしている事を考えると，問題 A は既によく見たコードのパターンであり，問題 B に比べるとコード理解度に差がつきにくかった事も考えられる．この推測の根拠としては，このコード A のみからは読み取れないキーボードの WASD による前後左右の移動が可能な事を回答中で述べている被験者が複数いた事が挙げられる．問題 A の得点に関しては，翻訳ありの場合は中央値が 80 点，翻訳なしの場合は 72.5 点と問題 B の場合と比較して大きな差はついていない．一方で，問題 B は翻訳なしの得点中央値が 27.5 点に比べ, 翻訳ありの場合は得点中央値が 80 点である．本翻訳システムの効果により問題 B のプログラムの内容についての理解度が上がったものと考えられる.

4.2 翻訳システムの主観的評価

被験者が主観的に本翻訳システムについての評価を行った結果は，13 人はポジティブ(翻訳ありのシステムがよい)と評価し，7 人はネガティブ(翻訳なしのシステムのほうがよい)となった．どちらでもない，と回答した被験者は 0 名で，ポジティブな評価とネガティブな評価が分かれる事になった.

図 4 を見ると，プログラム理解度，英語理解度が低い群が「翻訳あり」が良いと答える傾向にあり，プログラム理解度，英語理解度が高い群は「翻訳なし」が良いと答える傾向にある事がいえる.

アンケート自由回答を分析すると，「翻訳あり」が良いと回答したグループではその理由として「翻訳されたほうがわかりやすい」「直感的に理解できる」「理解する時間が少なくなって理解しやすい」「同時に見ると何のコードかわかりやすい」という理由を挙げている.

一方で「翻訳なし」のほうが良いと回答したグループは，「これまで見てきたソースは英語なので違和感」「書籍やネット上のソースは英語」「翻訳だと変数に誤解を招く表現がある」「日本語で書かれているとわからないものもある」「元々理解できるので英語しか見ない」のような理由を挙げた.

4.3 本翻訳システムの効果と今後の課題

図 4 および自由回答の結果を鑑みると，この翻訳システムはプログラムの理解度およびプログラムに利用される英語の理解度が低い群に特に効果があるのではないかと考えられる．今後実際に実装されるコード翻訳システムにおいては，コード翻訳はデフォルトでオフであるが，キー操作あるいはメニュー操作等によって起動し，元のコードと並列で翻訳コードを閲覧できるようなシステムにすることにより，必要な人が必要な時に利用できるシステムにする事がよいのではないかと考えられる.

特にプログラムの読解難易度が高いシステムで効果があるとすれば，他人が記述したコードやシステム等を分析する際には効果的な可能性がある.

また本研究結果から得られる重要な知見として, プログラミングの理解に英語の理解力が関与しているという事も付け加える. プログラム理解と英語理解には正の相関があり，コード翻訳がプログラム理解に有効であるケースがある事

から，日本人母語話者にとってみると英語がプログラム理解を妨げている可能性がある．その事も一因となり本コード翻訳システムがプログラムの内容理解を助けるツールとして機能していると言えるのではないかと考えられる．

本実験の被験者は, 東京工芸大学芸術学部ゲーム学科の学生であった．今回の実験でも元々のプログラム理解度によって差が出ているため，他の被験者ではどのような結果になるかわからない．また今回の実験では翻訳コード生成時のプロンプトで翻訳される範囲を限定して全てのコードが日本語に置き換わらないようにしているが，完全に日本語に置き換わった場合どのような効果があるかは不明である．例えば「int」「void」等の予約語は単純に日本語にするのではなく，「整数型変数」「帰値無型の関数」のように変換をしたほうが日本語としては可読性があがる可能性があるが，実際にはどのような変換をする事が日本人のプログラミング学習者にとって効果的かについてはわかっていない．今回の翻訳システムではコード内のコメントについては検証対象外としたが，ChatGPT を利用すればコードにコメントを付加する事は容易である．コメント翻訳との相互作用についても今後検証の必要がある．これらを含めてコード翻訳システムをどのように改良すればよいのかは今後の課題である．また本コード翻訳システムが Unity C#以外のプログラミング言語でも有効性がある事が期待されるが，他のプログラミング言語であっても同様の効果が得られるかは今後調査する必要がある．

5. おわりに

本研究では，日本人母語話者にとって英語理解の問題がプログラム理解に影響している事を仮説とし，ChatGPT を利用する事でコードの日本語翻訳と元のコードを並列で表記するシステムの有効性を検証した．

複雑度の高い Unity C#プログラムコードにおいてシステムの有用性を確認した．英語理解度とプログラム理解度には正の相関がある事もわかった．また主観での英語理解度が低くプログラム理解度も低いグループにおいて，本システムを高く評価した事を確認した．

プログラミング学習において, 学習者の母国語の問題はあまりフォーカスされる事がないが，本研究によりプログラミングで主に利用される英語への理解が日本人のプログラミング教育にとって重要である事を示せたと考える．

謝辞 本研究での実験にあたり，学生への告知等で協力を頂いた東京工芸大学ゲーム学科の教員諸氏にこの場をお借りして謝意を表明する．

参考文献

[1] Cwalina, K. and Abrams, B..Framework Design Guidelines: Conventions, Idioms, And Patterns for Reusable .NET Libraries, Addison-Wesley Professional, 2005
[2] Guo, Philip J. . Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018, p.1-14
[3] Guzman et al.. Experiences of Non-Native English Speakers Learning Computer Science in a US University. Proceedings of the 52nd ACM Technical Symposium on Computer Science Education. 2021, p. 633-639
[4] “highlight.js”. https://highlightjs.org/, (参照 2023-11-6).