単元間の依存関係に基づいた学習ロードマップ作成アプリの開発

学生証番号 153362　　情報理工学部　鎌田 弥利（荻原研究室）

1 はじめに

何かを学習する時、その分野の全体像ではなく、特定の話題や技術についてだけ学びたいことがある。しかし、一般の教科書や学習教材を提供するWebサイトを見ただけでは、どの部分をどんな順序で学べば良いのか、逆に、とりあえず学ばなくても良い部分はどこかということが必ずしも明らかではない。

本研究では、学習単元の依存関係を有向非巡回グラフ（DAG）としてモデル化し、必要な単元を適切な順序で学習できる学習セットを提供できるツールを試作した（図１）。

2 提案手法

すでに提供されている学習教材に対し、単元間の依存関係に考慮して学習コースを再構築する概念としてRimixがある[1]。

本研究ではRimixの概念とDAGを用いて、学習ロードマップ作成アプリを提案する。単元間の構造をDAGとしてモデル化し、トポロジカルソートを適用することで学習経路を導出する[2]。これによって、前もって学習しておくべき単元を順に並べることができる。

3 アプリの概要

提案するアプリは、課題データと依存関係データの２つのデータを読み込むことで動作する。２つのデータを読み込み、依存関係に基づいて課題データの順序を決定する。

学習したい目的の単元を選択すると、依存関係データから、その単元を学習する前提となる別の単元がわかる。このような依存関係のある単元を取り出して適切な順序で並べ、１つの学習セットにしたPDFファイルを作成できる。

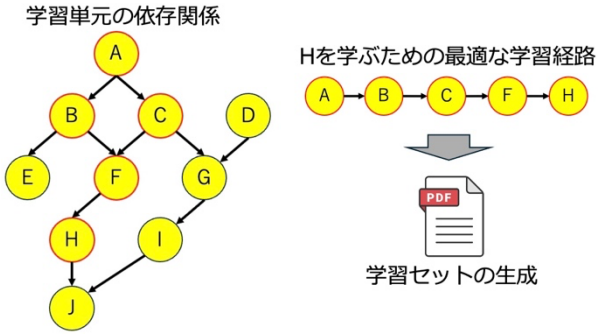

図１：アプリの概要

4 議論

本アプリにより、単元間の適切な学習順序を定めることができる。また、学習者が選択した単元を学ぶために必要な単元は必ず選択され、必要のない単元は学習セットには含まれない。このことから、学びたい内容に集中し、効率的な学習を行うことができると期待される。

しかし、本アプリはエラーハンドリングが不足していたり、読み込むデータの柔軟性に乏しいため、改善が必要である。

5 まとめ

本研究では、「学習の順序を適切に定めること」に重きを置き、トポロジカルソートによって学習ロードマップを作成するアプリを作成した。本アプリは、依存関係データを１つしか読み込むことができないが、複数読み込めるようにすることで、さらに複雑な依存関係の学習構造に対応できると考える。

参考文献

[1] LibreTexts: https://libretexts.org/
[2] 米澤，新藤: “学習支援のための学習課題関連付けナビゲーションシステム(2)”，情報科学誌フォーラム 2003.

動的なインタラクションを用いたウィジェット活用の可能性

学生証番号 153641　　情報理工学部　篠岡 尚輝 （荻原研究室）

1 はじめに

昨今、多種多様なスマホアプリが利用されており、雑多なホーム画面から目当てのアプリを見つけるのは手間である。そこで、ロック画面でも利用できるウィジェトを活用し、少ない手間でアプリの機能を利用する方法について可能性を探る。

本論文では、iOS ウィジェットにボタン操作を追加したアプリを例示し、利便性や効率の向上を目指す。

2 提案手法

本研究では、動的なウィジェットの一例としてiOSウィジェット上での動的インタラクションを活用し、アプリを起動せずに位置情報を記録・管理できる新しい仕組みを提案する。ウィジェット上のボタン操作で現在地を保存・分類し、最も近い地点の情報をリアルタイムに表示する機能を実装。App Intentを活用し、シームレスなデータ処理を実現する。これにより、ウィジェットの可能性を拡張し、利便性の向上を図る。

3 例示するアプリケーション

ウィジェットに簡単なボタン操作を追加することで、アプリ本体を開くことなく少ない手順で手軽にアプリの機能を使うことができる。図１を見ると従来のアプリにおいてユーザーに必要な操作は、スマホを取り出してからアプリを選択し、アプリの操作を行うところまでである。本提案ではスマホを取り出してからウィジェットをタップするだけで、あとはアプリ側の処理を一括に行える。

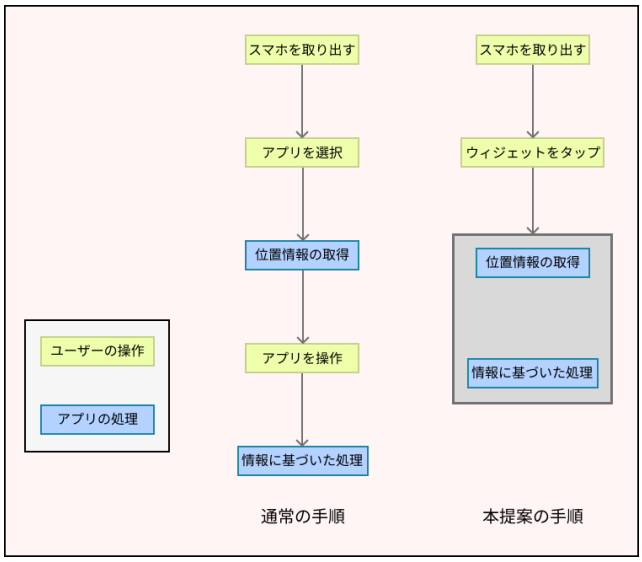

図 1　従来のアプリとの流れの比較

4 議論

ウィジェット上にボタンを追加することで、図１に示したように必要なステップを削減し、効率化を達成することができた。これは今回例に挙げたアプリに限らず、さまざまな形で応用が可能である。

しかし、動的なインタラクションの導入は、アクセス性向上やアプリ起動頻度の最適化につながる一方で、更新頻度の制約が課題となる。今後、ウィジェットの編集機能の活用や適切な更新条件の設定により、利便性の向上とリアルタイム性の強化が期待できる。

5 まとめ

本研究では、動的なインタラクションを用いたウィジェットを活用し、ユーザの利便性向上を図る。App Intentを用いてウィジェット上のボタンを実現し、直感的な操作を可能にしたが、表示スペースやバッテリー消費などの課題が残る。今後は応用範囲の拡大や操作性の向上を図り、ウィジェットの新たな活用方法を探る。

RPG の戦闘シーンにおけるバランス調整システム開発のための実験と考察

学生証番号 153704　　情報理工学部　鈴木 亮輝 （荻原研究室）

1 はじめに

本研究では文献[1] (以降「元論文」と呼称) に示された実験の再現を試み、実験の条件とパラメータの設定に関して考察を行なった。

元論文はコンピュータ (NPC) を相手とする簡単なターン制コマンドバトルのゲームを実験協力者にプレイしてもらい、ゲームバランスの自動調整について検討するものである。

プレイヤーと NPC は自分のターンで攻撃または回復を実行し、それに応じて体力が変化する（図１）。円内の数値は体力であり、攻撃力の値の何倍であるかを示している。

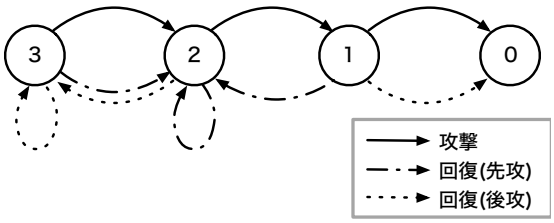

図１　状態遷移図

2 検討内容

元論文の実験では、対戦中に相手側の情報が開示されていない。相手の情報が見えている場合はどうなるのか検証が必要であると考えた。

また、元論文では体力の数値が攻撃力の値の 4 倍から 3 倍に変化するときに難易度が変化するという実験結果が示されている。しかし、他の場合については結果が示されていないため、再実験による検証を行う。

3 実験

本研究の実験ではプレイ画面 (図２) に相手側の体力、攻撃力、回復可能回数を表示する。また、プレイヤーの体力が攻撃力の 6 倍から１倍の場合の全てについて実験を行なった。

実験用のゲームをゲーム用統合開発環境のUnityで作成し、ゲームを無料で投稿、プレイできるサイトであるunityroomに投稿した。

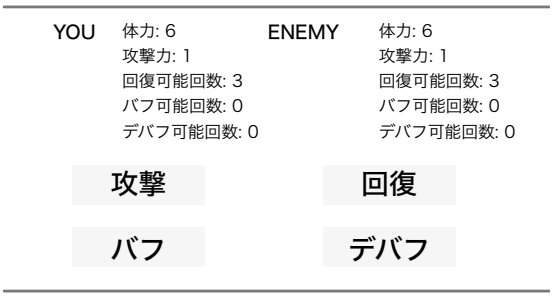

図２　ゲームのプレイ画面

投稿したゲームを実際にプレイしてもらい、プレイ後、数列として表示されたプレイ状況を見ながらアンケートに回答してもらった。

4 実験結果と議論

元論文と比較し、プレイヤーは体力が少ない状況でも「攻撃」を選択するように変化した。これは相手の情報を見て、自分の体力を減らしてでも攻撃すべきかの見極めができるようになったことが要因として考えられる。

体力が攻撃力の6倍から1倍の場合の勝率を比較すると、体力が少ないほど勝率は高くなるが、体力の値が4倍から3倍になった時には勝率は下がるという変化が見られた。ただし、元論文の結論と同一なのかは検証が必要である。

5 まとめ

元論文の実験方法に改良を加えて再実験を試み、元論文と類似した結果に加え、より多くの実験結果を得ることができた。また、プレイ内容の変化も観察することができた。

参考文献

[1] 高木幸一郎, 雨宮真人:“ロールプレイングゲーム(RPG)の戦闘におけるバランス自動調整システム開発のための基礎的考察”, 情報処理学会ゲーム情報学研究会, 5-5, 31-38 (2001).

パス文字列型を導入したUNIXシェルの試作

学生証番号 154613　　情報理工学部　吉田 奏瑠　（荻原研究室）

1 はじめに

Unixでは煩雑な繰り返し処理や定型的な手続きをシェルスクリプトで記述できる。しかしシェルスクリプトには一般のプログラミング言語とは異なる決まり事が多く、意図した通りに動作するスクリプトの記述は難しい。

その中でも、空白を含む文字列の扱いには特に注意が必要である。シェルは空白を区切り文字として認識するため、意図しない箇所で文字列が分割されてしまうからである。

空白を含む文字列の多くはファイル名やパスを表す文字列である。そこで本研究では、空白を区切り文字として扱わない文字列型の変数を備えたシェルの試作を行なった。これによって、利用者の直観により近い、平易な記述ができると期待される。

2 提案手法

本研究では、シェルスクリプトにおける変数および配列にパス文字列型を導入した。この型を用いると、値に含まれる空白は普通の文字として扱われ、値が空白によって分割されなくなる。これにより、空白によるエラーや予期せぬ動作を回避できる。

3 実装と動作例

実装は、オープンソースのシェル処理系[1]を改良して行なった。展開の際に空白の前にバックスラッシュ（\）を追加することで、空白で分割されないようにした。

図１にパス文字列型の構文を示す。従来のシェル変数および配列の構文と異なるのは代入記号（=>）のみであり、参照はこれまでと同様に行う（図２）。

パス文字列型の使用例を図３に示す。1行目で配列filesに、値に空白を含むファイル名を代入している。この代入に「=」を用いるとfilesは通常の文字列の配列となり、for文では空白で分割されるため、「a」と「1.txt」および「a」と「2.txt」が4行で表示されてしまう。空白で分割されないようにするには、3行目を次のように記述する必要がある。

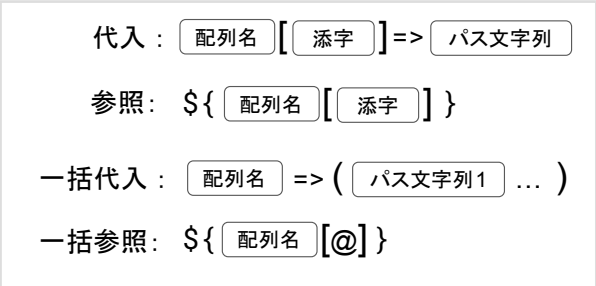

図1. パス文字列型の配列の構文

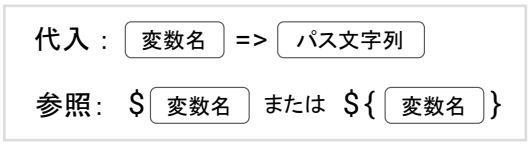

図2. パス文字列型の変数の構文

```
1: files=>("a 1.txt" "a 2.txt")
2:
3: for f in ${files[@]}  ; do
4:   echo $f
5: done
```

図3. パス文字列型の配列の使用例

```
for f in "${files[@]}"  ; do
```

一方、図３に示す通り、パス文字列型を用いたコードでは、ダブルクォートで囲まなくても、「a 1.txt」と「a 2.txt」が分割されずに１行ずつ表示される。

4 議論

現状ではパス文字列型の動作を確認しただけであり、スクリプトを実用的なレベルでは記述できない。例えば図３のようなfor文の場合、変数fはパス文字列型にはならない。このような記述にも対応できるように機能拡張を行うことが今後の課題である。

5 まとめ

シェルスクリプトの平易な記述を目的として、空白を区切り記号として扱わないパス文字列型を導入した。実用的な記述が可能かどうか、さらに検証を行う必要がある。

参考文献

[1]Ryuichi Ueda：“Rusty Bash”,
（https://github.com/shellgei/rusty_bash）