

C言語の理解を促進するRPG風ダンジョンゲーム作成に向けた マップ生成手法の提案

高橋 樂

京都工芸繊維大学

g-takahashi@se.is.kit.ac.jp

西浦 生成

京都工芸繊維大学

k-nishiura@kit.ac.jp

水野 修

京都工芸繊維大学

o-mizuno@kit.ac.jp

要旨

C言語は大学で学習する代表的なプログラム言語であるが、その習得は容易ではなく、C言語の学習意欲と理解を向上させるための研究は多く行われている。ゲーミフィケーションと呼ばれる、ゲームの要素やデザインをゲーム以外の分野に応用し、その分野に対するユーザの意欲を向上させる手法もその一つである。本研究では、C言語のソースコードの構造や動作を模したダンジョンを探検することでC言語の理解を助けるRPG風ダンジョンゲームの開発を目的とし、その実現に向け、ソースコードの内容に応じたゲームマップを自動生成するシステムを開発する。本システムではソースコードからフローチャートを作成し、その外形をマップとして反映させつつ、ワープなどのイベントやキャラクターを配置することでマップを作成する。本研究では制御構造や関数呼び出し、スカラー変数の宣言といったC言語の基本的な構文要素に焦点を絞ったシステム開発を行い、それらを反映したマップを作成できることを示した。

1. はじめに

C言語は大学において情報工学を専攻する学生が学習する代表的な言語である [1][2]。学生は、C言語の学習を通じてプログラム言語の基礎を学び、応用する力を身につける。しかし、プログラミング言語の初学者にとって、C言語は理解が難しいと言われている [3][4][5]。

また、ゲーミフィケーションとは、ゲーム以外の分野にゲームの要素を適用し可視化することで、その分野に対するユーザーの理解度と学習意欲を向上させる手法で

ある [6][7]。ゲーミフィケーションの効果はいくつかの研究で実証されている。Rogérioらは、言語学習のゲーミフィケーションであるDuolingoを用いて英語を学習した高校生が、用いずに勉強した生徒より成績が上昇したことを示した [8]。AlexandraとSherifは、健康活動のゲーミフィケーションであるFitbitが、慢性的な病気を患う子供の健康状態改善に好影響をもたらすことを示した [9]。

プログラミング学習におけるゲーミフィケーションはこれまでも提案されてきている [10][11][12]。例えば、谷本はクイズ形式でC言語プログラミングにおける問題を出し、正答数に応じてメダル等が得られるゲームを提案した [13]。しかし、これらの研究におけるゲーミフィケーションでは学習者が習得した知識の定着や応用を行う工程をゲーム化したものであり、初学者がC言語の構造や挙動を理解する工程をゲーム化しているわけではない。

そこで、本研究ではソースコードの処理内容を反映したマップを探索するRPG(ロールプレイングゲーム)風ダンジョンゲームの開発を目指す。C言語の構文要素をゲームに落とし込み、それらの要素の内容を視覚的かつ相方向的に表示することで、プログラムの構造や実行についての理解を深めることを目標とする。これにより、学習者はより短時間で理解できるだけでなく、C言語の新たな要素への学習意欲を高めることが期待される。プログラムを可視化した例は、ソースコードの一部を折り畳んで見やすくする [14]、処理をアニメーション化する [15][16]、データ構造を図示する [17]、デバッグとデータのアニメーションを一画面で表示する [18] 等がある。しかし、ソースコードをダンジョンゲームとして可視化した例は著者の知る限り存在しない。

この目的のために、本稿では特に、ソースコードからゲームに最低限必要な RPG マップを生成し描画するシステム（以降、マップ生成システムと呼ぶ）の開発に取り組んだ。まず、与えられたプログラムを静的解析することでフローチャートを構築する。その後、フローチャートのノードを部屋、エッジを通路やワープゾーンに置き換えることでマップの外形を表すデータを生成する。また、変数や関数に関する情報も抽出し、これらをマップ上のアイテムやイベントとして表現するためのデータを生成する。そして、これらのデータをマップ描画システムに入力することでマップを描画する。

本稿の以降の構成を示す。2 節では本稿で用いる用語を説明する。3 節ではマップ生成システムの目的を明確にする。4 節ではソースコードから RPG マップへの変換過程を説明する。5 節では提案システムによって生成されたマップの実例を C 言語の構文要素ごとに紹介する。6 節ではまとめと今後の課題を述べる。

2 用語

2.1 マップ描画システム

表示したいマップの情報を入力することで、ダンジョンゲームのマップを描画するシステムである。本システムは、本稿の最終著者が別の目的のために私的に開発したものを本研究に転用したものであり、一般的な RPG 風ダンジョンゲームを再現するための機能を有している。以下、本システムで描画できるマップの要素と機能についてそれぞれ説明する。

マップは、空間を表す正方形のマスと壁を表す正方形のマスが長方形型に並んだ領域である。

部屋は、マップ上における長方形の空間である。

通路は、マップ上における部屋と部屋を細い空間で繋いだものである。

パネルは、マップのマスを表す画像である。パネルは一意的な ID を持っており、この ID をマスに割り振ることでマスの内容を適切に描画できる。

キャラクターは、マップ上の空間のみを上下左右に移動する物体である。キャラクターにはプレイヤーキャラクター (PC) とノンプレイヤーキャラクター (NPC) の 2 種類がある。PC はユーザが操作するキャラクターで、移動させるほか NPC に話しかけるなどの行動を取れる。NPC はマップ上を自動的に移動する。

イベントは、マップ上に配置される移動しない物体で、プレイヤーが干渉することでゲーム内での変化を引き起こす。イベントの内容は自由に追加できる。

ワープゾーンは、イベントの一種で、指定された座標にプレイヤーを移動させる。

アイテムは、PC の所有物として表現される独立したパラメータである。マップが表示されているウインドウとは別のウインドウにアイテムの所持状況を表示させることができる。

2.2 抽象構文木

抽象構文木 (Abstract Syntax Tree, AST) は、プログラムの構文構造を表す木構造である。不要な構文情報を省略し、本質的な構造のみを保持するため、通常の構文解析木より簡潔となる。本研究では、C 言語の構文解析には Python のモジュールである Clang を用いた。

2.3 フローチャート

フローチャートは、処理の手順や判断の流れを有向グラフで表し、開始から終了までのプロセスを視覚的に示すためのものである。主に入力、処理、条件分岐、ループ、出力などの動作を明確にし、全体の構造や関係性を直感的に理解できるようにするために用いられる。本研究では、有向グラフの構築には Python のモジュールである Digraph を用いた。

2.4 A-star アルゴリズム

A-star アルゴリズムは、最短経路探索に用いられる探索アルゴリズムである。ダイクストラ法と貪欲法の原理を組み合わせることで、ある 2 地点の最適な経路を効率よく探索することができる。本研究では、このアルゴリズムを実装した Python のモジュールである astar を用いて、ダンジョンゲームのマップを構成するマス目状の 2 次元平面における 2 つの座標の最短の経路を取得する。

3 研究目標

3.1 本研究の実施範囲

本研究ではダンジョンゲーム作成の端緒として、C 言語のソースコードの構造を反映したゲームマップの自動

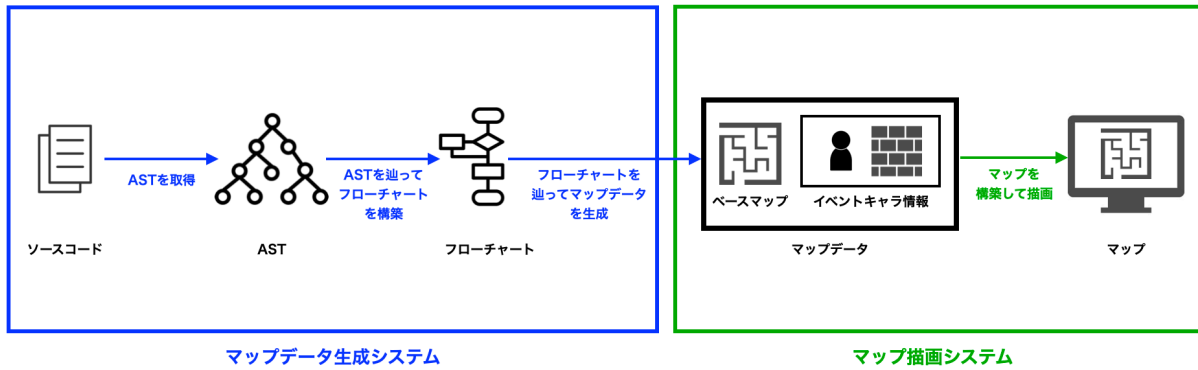


図 1: マップ生成システムの概要図

生成手法を開発する。また、段階を踏んで開発を進める都合上、本稿ではソースコードの基本的な構造の把握に必要な最低限の構文要素に限定してマップに反映させる。本稿で扱う構文要素は、(1) 逐次実行、分岐、ループを含む制御構造、(2) 関数の呼び出しと戻り、(3) スカラー変数の宣言、である。これ以外の、例えば配列や構造体、ポインタなどの特殊なデータ型や、変数の代入や演算といった複雑な処理は本稿では扱わず、今後の課題とする。また、本研究では単一のファイルに記述されたプログラムのみをマップ生成の対象とする。

3.2 マップの仕様

本研究の実施範囲において開発するマップ生成システムは以下を満たす必要がある。

- フローチャート上のノードで表されるひとつかたまりの処理のブロックを部屋に置き換え、エッジを通路もしくはワープ経路で置き換える。ブロックの行数に応じて部屋は広くなる。移動時間を削減するため可能な限り通路は短くする。また、通路が交差せざるを得ない配置となる場合は一方をワープ経路に置き換える。
- main 関数の開始地点となる部屋に PC を配置し、終了地点となる部屋にゴールを表す NPC を配置する。スタート地点からゴール地点までは必ず到達可能である。またフロー上、一方通行となる通路には逆戻りできない仕掛けを施す。
- 関数は、スタート地点とゴール地点が含まれる空間

とは独立した別の空間で表現する。関数の呼び出し地点には各関数に対応した NPC を配置し、PC が話しかけることで呼び出し先の関数にワープする。その際、関数の実引数に相当する変数をアイテムとして所持している必要がある。呼び出し先の関数を表す空間には戻りに対応するイベントが配置されており、PC がそこに到達すると呼び出し元の部屋にワープする。

- 変数は、PC が所有するアイテムとして表現する。グローバル変数は探索開始時点で所有しており、ローカル変数はソースコード中で宣言された位置に応じた部屋に配置された宝箱を開けることで所有できる。また、変数のスコープに応じてアイテムウインドウに表示されるアイテムが変化する。

4 ソースコードからマップへの変換

本研究で開発した、ソースコードからダンジョンマップを生成するシステムの概要図を図 1 に示す。本システムはマップデータ生成システムとマップ描画システムからなり、マップ描画システムには開発済みのものをベースに用いる。したがって、本研究の焦点はソースコードからマップ描画システムへの入力となるマップデータへの変換工程にある。以降では、ソースコードからフローチャートへの変換、およびフローチャートからマップデータへの変換についてそれぞれ説明する。

4.1 ソースコードからフローチャートへの変換

まずソースコードを構文解析することで AST を取得し、そこから必要な情報を抽出してフローチャートを構築する。構文解析には Python ライブラリである Clang の `cindex.parce` 関数を使用し、C11 準拠での構文解析を行うよう設定した。ソースコードから取得した AST に対して深さ優先探索を実施し、C 言語の制御構造に対応する構文要素である (1)if 文, (2)switch 文, (3)while 文, (4)do while 文, (5)for 文, に対応するノードを収集する。また、フローチャートの外形には反映されない情報として、(6) スカラー変数の宣言文, (7) 関数の宣言文, (8) 関数の呼び出し文, (9)return 文, に対応するノードを収集する。さらに、これらに当てはまらない式ノードの数を、フローチャートにおける各ノードにおける (10) 処理ブロックの大きさとして計測する。

フローチャートは AST のノード探索と並列して生成することができる。制御構造における分岐は AST の木構造と同一となるため、探索時の発見順にノードを生成し、直前のノードからエッジを繋げることで生成できる。一方、制御構造におけるループは AST の木構造には現れないため、ループの開始地点となるフローチャート上のノードを記録しておき、終了地点となるノードに到達した時点で開始地点へのエッジを生成する。main 関数およびその他の各関数についてそれぞれフローチャートを生成したのち、(6) から (10) の情報を各フローチャート中の対応するノードに追加する。

4.2 フローチャートからマップデータへの変換

マップ描画システムへの入力として、マップを構成するパネルの種類を二次元配列で記述したベースマップと、マップ上に配置されるキャラクターやイベントの情報が記述された設定ファイルが必要となる。まず、フローチャートを構成する有向グラフおよび付属情報をもとに部屋と通路を生成し、空間と壁からなる 2 値マップを生成する。その後、同様の情報を元にキャラクターやイベントを配置する座標を決定し、設定ファイルを作成するとともに、部屋の床や壁、イベントに応じたパネルの ID を二次元配列に格納することでベースマップを作成する。

部屋と通路の生成工程は次の通りである。まず、フローチャートの始点に近いノードから順に部屋を生成する。部屋の大きさはブロック内の連続した処理の数から

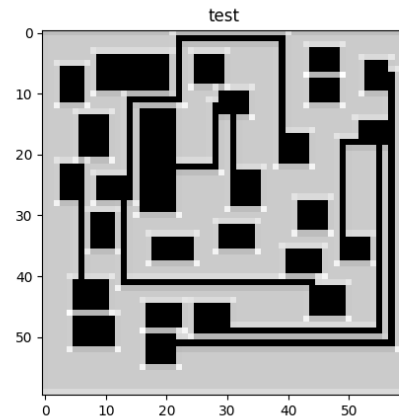


図 2: 生成したベースマップの例

決定される。生成される部屋は、他の部屋と隣接および重ならないような場所にランダムに配置される。このとき、現在のマップ全体の中に新しく部屋を配置できる場所がない場合は、マップ全体の大きさを右方向および下方向に拡張していく。新しく部屋を配置した際に、既に存在する部屋との間にフローチャート上でエッジを持つ場合に通路を生成する。2つの部屋の内部のランダムな座標を決定し、その2点間において他の空間に隣接しない最短経路を A-star アルゴリズムによって求め、経路上の壁を空間に変更することで通路を生成する。この際、条件を満たす経路が存在しない場合はワープイベントで代用する。

生成されたベースマップを可視化した例を図 2 に示す。黒い領域は探索可能な空間を、灰色の領域は壁を表している。壁の内部と部屋の壁面や四隅では別のパネルを使用しており ID が異なるため、可視化した際の色が異なっている。また、このベースマップをマップ描画システムに入力して描画したダンジョンマップの例を図 3 に示す。

5 生成されたマップの表示例

3.1 節で述べた本稿で扱う構文要素ごとに、開発したマップ生成システムによって各要素を含むソースコードから生成されたマップの例を紹介する。説明のためにマップ画像に部屋名とワープ経路を記入しているが、これらは実際の画面には表示されない。

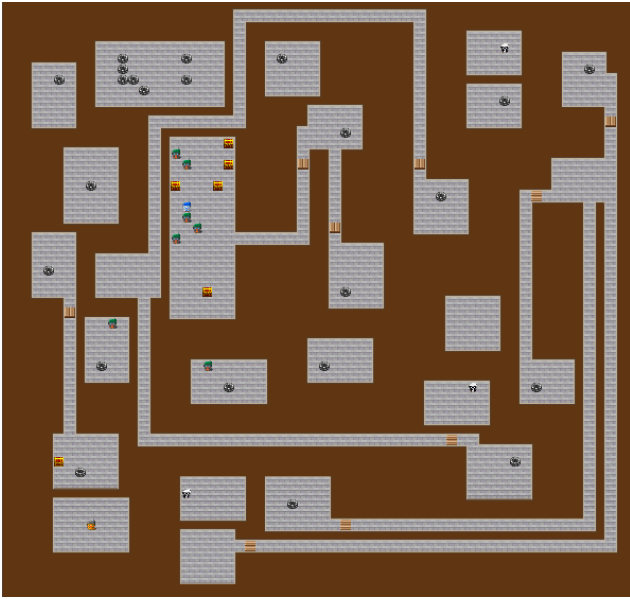


図 3: マップの全体像の例

5.1 if 構文の表現

図 4 に if 構文を含む次のソースコードを元にしたマップを示す。

```

1 int main(){
2     int i = 3, j = 4;
3     if (i == 5){
4         j *= 2;
5     }
6     else if(i < 5){
7         j++;
8     }
9     else{
10        j--;
11    }
12    return j;
13 }

```

部屋 S から部屋 A, 部屋 B, 部屋 C につながる 3 つの通路がそれぞれ if 構文の 3 つの条件分岐を表している。それぞれの通路の先には一方通行のパネルが配置されており、これは分岐先から分岐元の処理には戻れないことを表現している。元のソースコードではどの分岐先からも 13 行目の return 文に到達可能であり、生成されたマップでも、部屋 A の奥の通路を進むことでゴールキャラのいる部屋 G に辿り着くほか、部屋 B および部屋 C から到達できる部屋 D に配置されたワープイベントから部

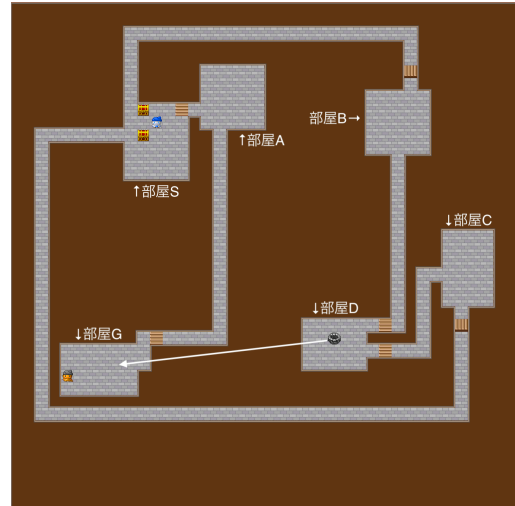


図 4: if 構文を表したマップ

屋 G に辿り着く。

5.2 switch 構文の表現

図 5 に switch 構文を含む次のソースコードを元にしたマップを示す。

```

1 int main(){
2     int i = 3, j;
3     switch(i){
4         case 1:
5             j = i;
6             break;
7         case 2:
8             j = i * 2;
9             break;
10        default:
11            j = i / 2;
12    }
13    return j;
14 }

```

開始地点となる部屋 S には 3 つのワープイベントが配置されており、それぞれが switch 構文における 3 つの分岐に対応している。遷移先の 3 つの部屋にもそれぞれ部屋 G へのワープゾーンが配置されており、元のソースコードの通りどの分岐からも同一の return 文に到達する。

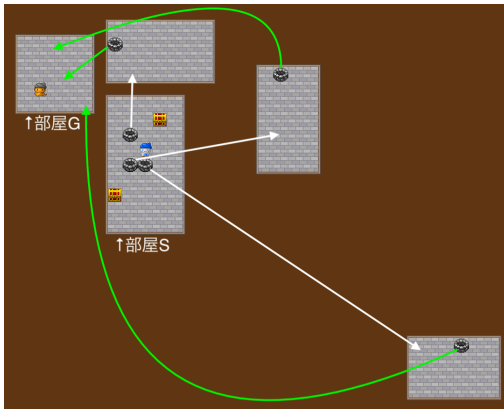


図 5: switch 構文を表したマップ

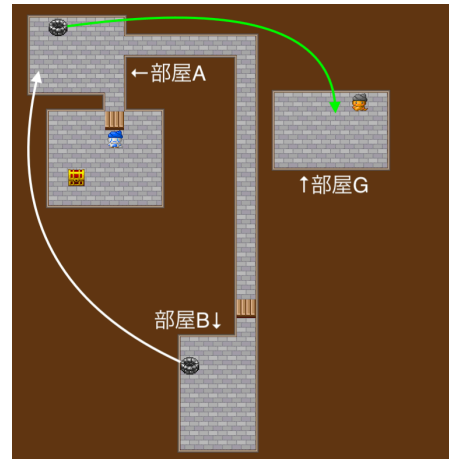


図 6: while 構文を表したマップ

5.3 while 構文の表現

図 6 に while 構文を含む次のソースコードを元にしたマップを示す。

```

1 int main(){
2     int i = 3, j = 4;
3     while(i--){
4         j += 2;
5     }
6     return j;
7 }
```

部屋 A が 3 行目の while 構文の条件式に対応しており、部屋 B に続く通路と部屋 G へのワープイベントを備えている。部屋 B は条件式が真の場合の処理に対応しており、配置されたワープイベントによって部屋 A に戻ることができる。このように部屋 A と部屋 B を行き来することで while 文によるループを表現している。一方、部屋 G へのワープイベントは条件式が偽の場合にループから外れることを表現している。

5.4 do_while 構文の表現

図 7 に do_while 構文を含む次のソースコードを元にしたマップを示す。

```

1 int main(){
2     int i = -3, j = -2;
3     do{
4         j++;
5     }while(i++);
6     return j;
7 }
```

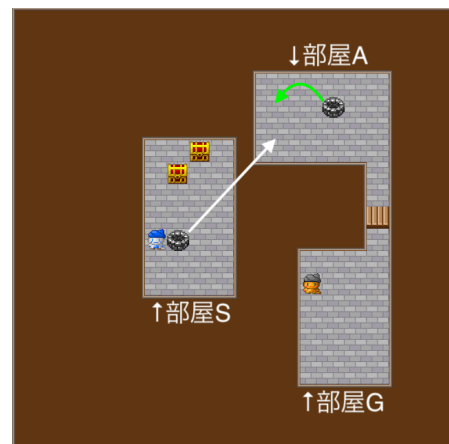


図 7: do while 構文を表したマップ

do_while 構文は while 構文とは異なり、先に繰り返したい処理を一度実行してからループさせる条件式を評価する。マップ上では、部屋 S が do 構文の中身の処理に対応しており、部屋 S からワープできる部屋 A が条件式に対応している。部屋 A は条件式が真の場合に部屋 S に戻るループ構造と、条件式が偽の場合にループから外れる部屋 G への通路を備えている。

5.5 for 構文の表現

図 8 に for 構文を含む次のソースコードを元にしたマップを示す。

```

1 int main(){
2     int i, j = 4;
3     for (i = 0; i < 5; i++){
```

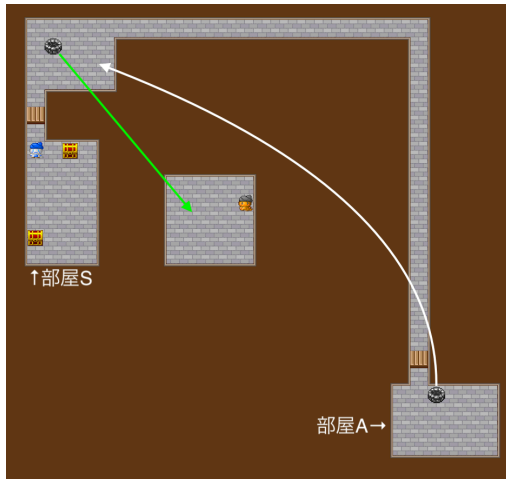


図 8: for 構文を表したマップ

```

4     j *= 2;
5   }
6   return j;
7 }

```

for 構文を反映したマップの構造は図 6 で示した while 構文のマップと同様である。本稿のシステムではまだ演算式をマップに反映できておらず、実際には変数 i の初期化 ($i = 0$) の式をループ開始前の部屋 S に、また変数 i の変化 ($i++$) の式はループ内処理の最後に毎回実行されることを反映した部屋 A に配置されることを意図したマップとなっている。

5.6 スカラー変数の表現

図 9 にスカラー変数の宣言を含む次のソースコードを元にしたマップおよびアイテムウィンドウを示す。

```

1 int k = 3;
2 char l;
3 float a = 0.2;
4
5 int main(){
6   int i = 3, j = 4;
7   j++;
8   l = 'b';
9   return j;
10 }

```

部屋の 2 つの宝箱は、6 行目の main 関数内で宣言されたローカル変数 i , j をアイテムとして入手できるイベントを表している。これらの宝箱を開けることで、図 10



図 9: スカラー変数の宣言を表したマップ



図 10: アイテム（変数）の取得

のようにアイテム i , j を取得し、アイテムウィンドウに白文字で追加される。

また、メイン関数の外側で宣言されたグローバル変数は初期状態でアイテムとして所有している。グローバル変数アイテムはローカル変数アイテムと区別するために緑色で表示されている。また、グローバル変数 k および a のようにプログラム内で一度も参照されないグローバル変数はアイテムウィンドウに表示されない。

5.7 関数の表現

図 11 に関数を含む次のソースコードを元にしたマップを示す。

```

1 int g1 = 3;
2
3 int funcA(void){
4   int c = 2, d = 3;
5   c++;
6   d--;

```

```

7   g1 = 5;
8   return c * d;
9 }
10
11 void funcB(int i, char j);
12
13 int main(){
14     int i = 3;
15     char h = 4;
16     h++;
17     funcB(i, h);
18     return funcA();
19 }
20
21 void funcB(int i, char j){
22     int h = i * j;
23     h--;
24 }

```

関数は main 関数に対応する空間とは別の空間として表現され、関数の呼び出しは NPC を介したワープイベントとして表現されている。部屋 S にはゴールキャラの他に 2 人の NPC が配置されており、右側の NPC は部屋 A への関数呼び出しに対応している。ワープしたい関数の実引数に対応する変数のアイテムを所持している場合に限り、PC がこれらの NPC に話しかけることで対応する部屋にワープできる。

例えば、11 行目に示すように部屋 B に対応する関数 funcB は 2 つの実引数 i, h を持っており、これらの変数は main 関数内で宣言されている。PC がこれらの変数に対応する全てのアイテムを宝箱から入手する前に部屋 B へのワープを行う NPC に話しかけた場合、図 12 のようにワープができない旨と不足しているアイテムの情報が表示される。一方、必要なアイテムを有している場合には部屋 B にワープする。その際、遷移先の関数のスコープに応じてアイテムウィンドウに表示されるアイテムが変更される。この場合では、図 14 に示すように、実引数である main 関数ローカル変数 i, h が関数 funcB の仮引数である funcB 関数ローカル変数 i, j に変換されている。また、部屋 A に遷移する場合は対応する関数 funcA に仮引数が存在しないため、図 13 に示すようにグローバル変数のみが表示されている。

呼び出される関数内の return 文がある部屋にも NPC が配置され、PC が話しかけることで、最後にこの関数へのワープを行なった NPC の元にワープすることがで

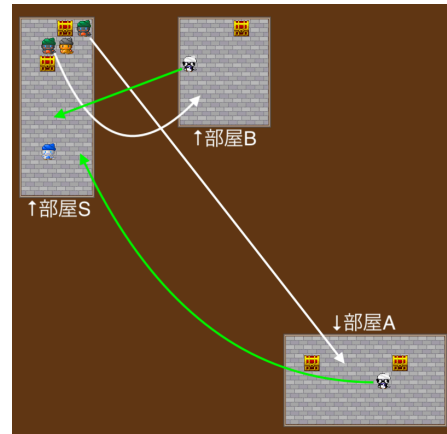


図 11: 関数を表したマップ



図 12: 関数 funcB への遷移条件を満たしていない場合

きる。この様子を図 15 に示す。この場合も関数へのワープの際と同様に、呼び出し元の関数のスコープに応じてアイテムウィンドウに表示されるアイテムが更新される。

6. おわりに

本研究では、C 言語のソースコードを反映したダンジョンゲームの内、プログラムフローやスカラー変数、関数の呼び出しと戻りだけを反映した簡素なマップを生成するシステムを開発した。その結果、部屋間の遷移やキャラクター等を通じて、それらの C 言語の要素をマップ上で表現することに成功した。

しかし、配列やポインタなどの C 言語特有のデータ型や、変数の演算式など、まだ表現できていない要素も多い上に、複数のファイルからなるプログラムのマップを生成することはできていない。また、プレイヤーがキャラクターを操作してゴールを目指すだけのゲーム性に留まっており、ユーザーの C 言語学習のモチベーションを

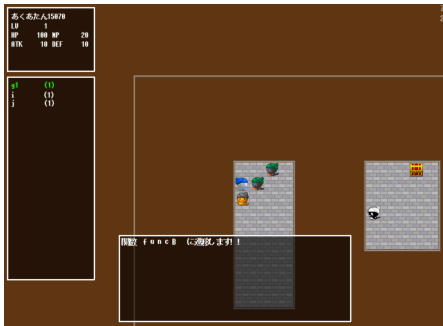


図 13: 関数 funcB の部屋に遷移



図 14: 関数 funcA の部屋に遷移

高めるには乏しい。今後、これらの要素の実現とゲーム性の拡張を行うことで、様々なC言語のソースコードプログラムを詳細にゲームマップ化でき、学習者のC言語の理解度とモチベーションの向上を実現できるゲームの開発に取り組みたい。

参考文献

- [1] Richardson and Jeffrey J. Effectively teaching c in an introductory embedded microcontroller course. In *Proceedings of II/IN Sectional Conference, Illinois*, pp. C-T1-4, 2005.
- [2] Xiaoli He, Yu Song, Yuxin Du, Yongming Huang, Xiaodong Yin, and Xia Long. Research on teaching reform of c programming language course for deep integration of professional fields. *Open Journal of Social Sciences*, Vol. 10, No. 1, pp. 267-275, 2022.
- [3] Liping Deng, Xianfang Zou, and Wenjuan Hu. Pointer teaching of c language in higher voca-
- [4] Dongkyun Lim, Ji-Eun Lee, Gijun Um, and Dosik Moon. Development of a c language education instructor training course using animated characters. *International Journal of Engineering & Technology*, Vol. 7, No. 3.33, pp. 41-45, 2018.
- [5] Shukun Liu, Zhen Chen, and Jinpeng Tang. The improved methods of teaching practice based on c language programming. In *Proceedings of the 2013 Conference on Education Technology and Management Science (ICETMS 2013)*, pp. 807-810. Atlantis Press, 2013.
- [6] Awaz Naaman Saleem, Narmin Mohammed Noori, and Fezile Ozdamli. Gamification applications in e-learning: A literature review. *Technology, Knowledge and Learning*, Vol. 27, No. 1, pp. 139-159, 2022.
- [7] Ana Manzano-León, Pablo Camacho-Lazarraga, Miguel A Guerrero, Laura Guerrero-Puerta, José M Aguilar-Parra, Rubén Trigueros, and Antonio Alias. Between level up and game over: A systematic literature review of gamification in education. *Sustainability*, Vol. 13, No. 4, p. 2247, 2021.



図 15: 関数 funcB から呼び出し元に戻る場合

tional colleges. In *Proceedings of the 2016 International Conference on Advances in Management, Arts and Humanities Science*, pp. 438-441. Atlantis Press, 2016.

- [8] Iris Cristina Peláez-Sánchez and Anabel Velásquez-Durán. The impact of duolingo in developing students' linguistic competence: an aspect of communicative language competences. *Educação e Pesquisa*, Vol. 49, p. e252467, 2023.
- [9] Alexandra M Kasparian and Sherif M Badawy. Utility of fitbit devices among children and adolescents with chronic health conditions: a scoping review. *Mhealth*, Vol. 8, pp. 1–17, 2022.
- [10] Zehui Zhan, Luyao He, Yao Tong, Xinya Liang, Shihao Guo, and Xixin Lan. The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, Vol. 3, p. 100096, 2022.
- [11] Rania Elshiekh and Laurie Butgerit. Using gamification to teach students programming concepts. *Open Access Library Journal*, Vol. 4, No. 8, pp. 1–7, 2017.
- [12] Margarita Elizabeth Ortiz Rojas, Katherine Chiluiza, and Martin Valcke. Gamification in computer programming: Effects on learning, engagement, self-efficacy and intrinsic motivation. In *11th European Conference on Game-Based Learning (ECGBL)*, pp. 507–514. Acad Conferences LTD, 2017.
- [13] 谷本嵩晃. ゲーミフィケーションを使用したc言語学習支援モバイルアプリケーションの開発. Master's thesis, 京都工芸繊維大学, 京都府京都市左京区松ヶ崎橋上町, 2 2024.
- [14] Devamardeep Hayatpur, Daniel Wigdor, and Haijun Xia. Crosscode: Multi-level visualization of program execution. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2023.
- [15] Carlos R. Jaimez González and Miguel Castillo-Cortes. Web application to support the learning of programming through the graphic visualization of programs. *Int. J. Emerg. Technol. Learn.*, Vol. 15, pp. 33–49, 2020.
- [16] Alvin Prasad, Kaylash Chaudhary, and Bibhya Nand Sharma. Programming skills: Visualization, interaction, home language and problem solving. *Education and Information Technologies*, Vol. 27, pp. 3197–3223, 2021.
- [17] Imre Bende. Data visualization in programming education. *Acta Didactica Napocensia*, pp. 52–60, 2022.
- [18] Elvina Elvina, Oscar Karnalim, Mewati Ayub, and Maresha Caroline Wijanto. Combining program visualization with programming workspace to assist students for completing programming laboratory task. *Journal of Technology and Science Education*, pp. 268–280, 2018.