卒 業 研 究 報 告 書

題　目　　ソフトウェアリポジトリから
ロールプレイング風ゲームを生成するツールを
利用したソースコードメトリクスの可視化

指導教員　　水野　修　教授

崔　恩瀞　助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　16122004

氏　　名　　上北　裕也

令和2年2月12日提出

ソフトウェアリポジトリからロールプレイング風
ゲームを生成するツールを利用したソースコードメトリクスの可視化

令和 2 年 2 月 12 日

16122004　上北　裕也

概　要

ソフトウェアの可視化は，ソフトウェアリポジトリマイニングを促進する手法の一つである．ソフトウェアの特徴を示すメトリクスの適切な可視化が行われることにより，ソフトウェアへの理解が促進され，より詳細にソフトウェア内部を分析する動機づけを与えることができる．

中でもインタラクティブなソフトウェアの可視化はこういった動機づけに適していると考える．既存のインタラクティブなソフトウェア構造可視化手法の多くは 3 次元オブジェクトを利用したものであったが，これらは描画負荷が高い．また，VR（Virtual Reality）機器は未だ一般に普及しておらず導入しづらい．当研究室では先行研究として既にリポジトリ探訪者がソースコード内をインタラクティブに探索できるウェブアプリケーション「GitQuest」が存在する．これは，Git リポジトリのディレクトリからロールプレイングゲーム風 2 次元マップを，ファイルからノンプレイヤーキャラクターを生成するものである．本研究ではこの GitQuest を基に 2 次元表現を用いてインタラクティブな可視化を行う事によりユーザーが直感的にソースコード評価を行える可視化手法を提案する．

具体的には「GitQuest」にリポジトリ内のソースコードメトリクスを取得する機能及びそれに応じて生成マップに変化を加える機能を追加した．実装としては，サーバーに用意された WebAPI よりソースコードメトリクスを取得し，メトリクスに応じてキャラクターのグラフィックや，マップオブジェクト数を変更した．

生成されたマップにソースコードメトリクスを反映させる事により，ユーザーに対してソースコードの複雑さを視覚的に表現できる．これにより，ユーザーに対してソースコード評価の際，複雑なファイルの多いディレクトリを視覚的に印象付けできる．

目　次

1. 緒言

ソフトウェアリポジトリマイニングを促進する手法の一つに，ソフトウェアの可視化がある．ソフトウェアの特徴であるメトリクスを適切に可視化する事によってソフトウェアへの理解を促進し，より詳細なソフトウェア内部の分析を促す動機づけを与える．

中でもインタラクティブにソフトウェアを可視化する研究は，ソフトウェア内部分析の動機付けに対して大きな効力を持つと考える．多くのインタラクティブなソフトウェア可視化研究の共通点として，モジュールは3次元オブジェクトを用いて可視化されていることが挙げられる．3次元オブジェクトを用いる場合，描画負荷が高くなりスマートフォンのブラウザなどで動かしづらくなる．またバーチャルリアリティを用いたものは，準備にかかる費用などコストが高い．2次元のソフトウェアメトリクス可視化ツールも既に開発されてきたが [1]，インタラクティブにソフトウェアを探索することはできない．2次元のインタラクティブなソフトウェア可視化ツールを用いる事によって，開発者以外のユーザーに「テレビゲーム的な直感操作」及び「すぐ始められる気軽さ」を提供するソースコード探索ができるのではないかと考える．

当研究室の先行研究として既にリポジトリ内をインタラクティブに探索できるウェブアプリケーション「GitQuest」が存在する．これはバージョン管理システムである Git[2] の管理しているリポジトリからロールプレイングゲーム風のマップを生成し，その中を自由に探索できるというものである．本研究はこの「GitQuest」のマップにリポジトリ内のソースコードのソースコードメトリクスを反映させる機能を追加するものである．これによって，ユーザーが直感的かつ気軽にソースコード評価を行える可視化手法を提案する．

最後に本書の以降の構成を示す．まず，第2章で本研究の目的を述べる．第3章では本研究のベースとなった「GitQuest」についてその基本機能を述べる．第4章ではソースコードメトリクスの可視化にあたって，可視化対象，表現方法，実装方法を順を追って説明する．第5章では本研究よりもたらされた考察について述べる．第6章では本研究の関連研究について述べ，最後に第7章で本研究の結言を述べる．

2. 目的

2.1 本研究の目的

本研究の目的は，ユーザーが直感的かつ気軽にソースコード評価を行える可視化手法の提案である．特に本研究においてはリポジトリ内のソースコードメトリクスを，専門知識を持たないユーザーに対して直感的に伝わるような実装を目標とする．

2.2 ロールプレイングゲーム風インターフェースを利用する狙い

この実現のために，本研究ではロールプレイングゲーム風インターフェースを用いる．ゲームのジャンルについて明確に定義するのは難しい事であるが [3]，本研究においてのロールプレイングゲームは，主人公がなんらかの目的のために敵との戦闘や難解なダンジョンなどの困難を退け，自身の能力値や装備などを成長させる事によって目的を達成するものを想定する．

ゲーム風インターフェースを用いるのは，本来リポジトリ内の探索に伴う煩わしさの削減及びリポジトリ内の探索に対する抵抗の減少を狙ってのことである．しかし，中でもロールプレイングゲーム風インターフェースを利用する狙いは，メトリクスを多岐にわたる方法で可視化する事にある．

本研究ではファイルをキャラクターとして表現しているが，多くのロールプレイングゲームではキャラクターに「性別」「年齢」「種族」「体長」「ステータス」といった特徴がある．故に適切な実装を行えば，ソフトウェアメトリクスを数種類取得した場合，これら１つ１つにメトリクスを反映させることができる．このため，ロールプレイングゲーム風インターフェースを利用することで多くのソースコードメトリクスを取得しながらその多くをキャラクターの特徴として可視化できると考え，この実装に踏み切るに至った．

3. GitQuest

本章では，先行研究の GitQuest について述べる．

3.1 仕様

GitQuest は Git のリポジトリのデータを基に，ディレクトリからマップ，ファイルからノンプレイヤーキャラクターを生成し，ユーザーがプレイヤーとしてマップ内をインタラクティブに探索できるウェブアプリケーションである．

プレイヤーは画面中央のキャラクターを操作して，マップ内を探索する．操作感覚は一般的なコンピューターロールプレイングゲームを踏襲しており，十字キーで移動する．この時マップサイズが画面より大きければ，画面端に行くとマップ全体のサイズに応じてスクロールを行う．また，Space キーで他のキャラクターに対してアクションが実行できる．プレイヤー以外のキャラクターはリポジトリのファイルに対応しており，アクションを実行すると該当するファイルの名前を表示する．

マップは木々や段差によって移動が制限されている．これは純粋なリポジトリ探査ツールとしては不便な点であるが，コンピューターロールプレイングゲームの踏襲としてあえて残したものである．図 3.2 に示すマップの随所に設置されたゲートはディレクトリのゲーム内表現であり，くぐると該当するディレクトリのマップへ移動する．移動先ディレクトリではまた新規に地形が生成されており，ゲート，キャラクターに加え上層のディレクトリに帰る BACK ゲートが存在する．こちらのゲートの画像は図 3.3 に示す．なお先行研究の段階ではマップサイズ以外のキャラクター，オブジェクトの配置を含めたマップの地形は常にランダムに生成される．

ホストアドレスに”/#/user/reponame”と続けてリポジトリを指定してサイト内遷移することで対応するリポジトリのマップが表示される．ゲートなどでディレクトリ移動が起こると現在のマップとそこに配置されたオブジェクトを全て消去し，移動先のリポジトリ情報を得て新たなマップを生成する．

図 3.1　GitQuest

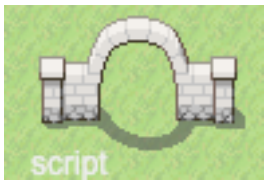

図 3.2　ディレクトリのゲーム内表現であるゲートのグラフィック

図 3.3　BACK ゲートのグラフィック

3.2 実装

GitQuest は JavaScript のフロントエンドフレームワークである，React[4] を用いて制作されている．

リポジトリ情報の取得は GitHub API[5] を用いて行われる．ページの更新，ディレクトリの移動などが行われると GitHub API からユーザー名，リポジトリサイズ，ファイル名，ディレクトリパスなどのデータを取得し，そのデータからマップ，キャラクターを生成する．

本研究でも，基本的にこれを流用する．次章以降では，追加で実装する機能について説明する．

3.2.1 React

React は JavaScript のフロントエンドフレームワークである．ユーザーインターフェースを構築するための宣言型ライブラリであり,「コンポーネント」と呼ばれる小さく孤立したコードから複雑なユーザーインターフェースを構成できる．GitQuest においては URL からのルーティングに使用する．

3.2.2 CreateJS

CreateJS は，HTML5 を利用したリッチコンテンツ制作用のソフトウェアパッケージである [6]．主に 4 つのライブラリによって構成されており，GitQuest においては HTML5 Canvas[7] での制作を扱うための JavaScript ライブラリである EaselJS[6] を主に利用する．これは，ゲーム画面の描画に HTML5 Canvas を用いるからである．

4. ソースコードメトリクスの可視化

本章では，本研究の追加した機能として，3.1 節で述べた GitQuest 内オブジェクトにメトリクスを反映する機能について述べる．

4.1 可視化の対象

可視化するメトリクスを表 4.1 に示す．ソースコード行数（以下 LOC）と関数数（以下 Functions Of Code，FOC）はソースコードの特徴を表すものとして用い，リポジトリサイズ，ディレクトリ深度，ファイル数はディレクトリの特徴を表すものとして用いる．

LOC を対象とする理由はソースコードメトリクスとして非常にシンプル [8] なものであり，ソースコードメトリクスとして代表的なものだからである．LOC のみを用いてソースコードを評価すると LOC の定義によって偏りが存在するため [8]，FOC も併用する事で妥当性を保つ．

4.2 GitQuest における表現

GitQuest において以上のメトリクスを表現するにあたり，視覚的に変化を伝えるためグラフィックにメトリクスを反映させる事にした．

4.2.1 LOC，FOC

ファイルに対応するキャラクターグラフィックにソースコードの規模（以下コード規模）を反映する．このコード規模は LOC と FOC を用いて以下の式 4.1，4.2 によって表される．表 4.2 に示すようにコード規模が一定の閾値を超えるごとにグラフィックカラーを変更する．実際のグラフィックを図 4.1 に示す．

$$k = 1 + LOC + FOC * 10 \tag{4.1}$$

$$K = 2 * \log_{10}(k) \tag{4.2}$$

表 4.1　取得するメトリクス一覧と取得理由

取得メトリクス	取得理由
LOC	ソースコードメトリクスとして非常にシンプル，ソースコードメトリクスとして代表的なもの
FOC	LOC の定義によって生じるメトリクスの差を縮小するため
ファイル数	広いマップでファイルの存在を示す指標を作るため
リポジトリサイズ	マップサイズの決定に用いる
ディレクトリ深度	ディレクトリの深さを示す指標を作るため

ただし, K をコード規模とし, K の小数点以下は切り上げる. LOC が 10000 以上になるなどコード規模が 9 以上になった場合は未実装である.

4.2.2 ファイル数，リポジトリサイズ

先行研究では，リポジトリサイズに応じてマップサイズが変化するようになっていた．ファイル数とキャラクター数が対応しているのは 3.1 節でも述べた通りである．しかし，リポジトリサイズが大きい場合，マップが広く一見画面内にキャラクターが存在しない場合がある．ファイルが存在することを示すため，ファイル数が多く，サイズの大きいディレクトリには図 4.2 に示す目印となる建物のオブジェクトを配置する事とした．これを以降建物オブジェクトと呼称する．ファイル数，リポジトリサイズ，ディレクトリ深度によって式 4.3 で算出される個数分建物オブジェクトを配置する．

$$B_{num} = \log_{10}(F_{num} + 1) * (\frac{R_{size}}{3} - depth) \tag{4.3}$$

ここで, B_{num}は建物オブジェクトの個数, F_{num}はファイル数, R_{size}はリポジトリサイズ, $depth$ はディレクトリ深度である．計算結果が 0 以下の場合はオブジェクトを配置しない.

4.2.3 ディレクトリ深度

ディレクトリが深くなった場合，生成されるマップ中の木々のオブジェクト（木々オブジェクト）を増やす．また，ディレクトリ深度が一定を超えるとタイルセットを変え，マップの雰囲気を変える．これについては 4.3.3 項で述べる．

4.2.4 マップの固定化

なお 3.1 節で述べたように，先行研究の段階では生成されるマップがサイトを訪れるたびにランダムに生成されていた．メトリクスの反映が分かりづらいこと，どのリポジトリにいるか覚えづらいことなどからメトリクスの反映前にこれを修正し，同一のディレクトリに対しては同一のマップが生成されるようにする.

表 4.2　コード規模と色の対応

色	対応するコード規模	FOC が 0 の時の対応 LOC
黒	0	0
紫	1	1〜3
青	2	4〜10
水色	3	11〜31
緑	4	32〜100
黄緑	5	101〜316
黄	6	317〜1000
桃	7	1001〜3162
赤	8	3163〜10000

図 4.1　メトリクスによる色変化　左から黒，紫から赤

図 4.2　建物オブジェクト

4.3 実装

4.3.1 ファイルメトリクスの取得，反映

実装を行うにあたり，Java のソースコードを対象にメトリクスの取得を行うこととした．GitHub API では LOC などのソースコードメトリクスは取得できなかったため，リポジトリからソースコードメトリクスを取り出す API を用意し，起動時に通信を行う事によって JSON 形式のデータを得る．実際にデータを取得する方法については節で述べる．このデータを元にファイルメトリクスを反映させる．メトリクスを取得できない Java 以外のファイルは一律黒色となる．

4.3.2 オブジェクト生成

4.2.2 項で述べたように建物のオブジェクトを目印に用いた．リポジトリサイズは GitHub API より取得可能であり，ファイル数はマップ生成時に確認可能である．この時，建物オブジェクトは木々オブジェクトより先に配置する．

先行研究のオブジェクトの配置方法は以下の通りある．

1. マップのランダムな位置を指定する
2. 配置可能であれば配置，そうでなければ位置指定からやり直す．
3. やり直しが規定回数に達した場合，オブジェクトを配置せずに処理を終える．
4. これをオブジェクトの数だけ行う．

配置できるオブジェクトの数は限りがあり，後の方に配置されたオブジェクトは配置されない可能性がある．建物オブジェクトは木々オブジェクトより重要度が高いため，先に配置することで喪失が起きにくいようにする．

4.3.3 ディレクトリ深度

ディレクトリ深度はリポジトリのルートディレクトリを 1 とし，一つ階層を経る事に 1 増える．深度が 3 を超えるとグラフィックが図 3.1 に示す平原のものから図 4.3 に示す山用のものに変更される．また，山の雰囲気を出すためディレクトリ深度とマップサイズに応じて木々オブジェクトの配置個数を増やす．具体的なオブジェクトの配置については 4.3.2 項で述べた通りである．

図 **4.3**　ディレクトリ深度 **3** 以降のマップグラフィック

4.3.4 マップ固定化

これは，マップ生成に使う乱数に初期乱数を指定する事によって実装を試みた．しかし JavaScript の Math.random メソッドは初期乱数を設定することができない．これは，外部の乱数生成ライブラリ seedrandom[9] を用いることで解決した．ルートディレクトリからのパスから初期乱数を生成する事によって，同じディレクトリに対しては常に同じマップを生成する．

4.4 メトリクスの取得

4.3.1 項においてメトリクスの取得について述べたが，本章ではその具体的な方法について述べる．

4.4.1 メトリクスをクライアントで持たない理由

本研究ではメトリクスをクライアントで持たず，サーバー上の Web API を利用してメトリクスデータを取得している．この主な理由はデータ量の問題である．クライアント側でこの処理を行うとデータ量が多くなり，読み込み時間が多くなってしまう．また，クライアントが持つリポジトリデータが限られるため探索できるリポジトリが限られてしまうという問題もある．サーバーからデータを取得する事によって，これらの問題を解決する．

4.4.2 LOC，FOC の定義

LOC，FOC を取得するにあたって，その定義が必要となる．本研究における LOC は，コメント，空行を含む全ての行を LOC として扱う．FOC はクラスの数を関数数とする．

4.4.3 メトリクス取得コード

LOC と FOC の計算に用いるコードについて説明する．コードは Python™[10] を利用して書かれており，リポジトリ内のファイルのメトリクスを取り出してデータベースファイルに格納する．LOC はファイルを先頭から辿り，各行を経るたびにカウントすることで求める．FOC はコード中のクラスの定義数を数え，それをそのま

ま FOC として扱う．取得したメトリクスは，サーバ計算機のデータベースに格納される．このデータにブラウザ上の Javascript からアクセスするために，簡易 Web API を作成し，データの取得を行う．

5. 考察

5.1 本研究の成果

本研究によって，コード規模の大小がグラフィックに反映された．これによって，プレイヤーは規模の大きそうなファイルの多いディレクトリを視覚的に知ることができる．例えば，赤色のキャラクターばかりいるマップは，コード規模が大きい複雑なソースコードが集まったディレクトリであることがわかる．ユーザーはこのようなディレクトリについてその重要性に気づくことができ，視覚的にも印象付けられる．

5.2 今後の目的

現状ではファイルメトリクスが可視化されているだけであり，ツールとしての機能性はまだ開発段階である．

当研究の最終的な目標は，プログラミング知識の有無にかかわらずリポジトリ内の探索及びソフトウェア工学に役立つ知見の収集を行えるウェブアプリケーションの開発である．具体的な目標として本ソフトウェアをゲーム化し，ゲームの難しさとソフトウェアの複雑さの対応をモニタすることがある．実際にゲームをプレイする際にプレイが進まなくなる等のボトルネックが存在することが想定されるが，そうした部分がソフトウェアの複雑さにおいても危険な部分として検出されるような仕組みを考えたい．

5.3 今後必要とされる機能

前節の目的を達成するため，今後当アプリケーションに以下のような機能の追加が必要になる．

1. 文法エラー表示機能

   リポジトリ内のソースコードの文法エラーなどを検出する機能．当アプリと並行して静的解析器を用い，エラーを検出した場合，該当ファイルがどのようなエラーを含んでいるかがひと目でわかるようにする．

2. より高度なメトリクス取得，反映機能

今現在ではソースコードメトリクスとしてLOCとFOCのみを使用しているが，更に妥当性を上げるため他のメトリクスを取得する．取得したメトリクスはグラフィックの大きさや形など別の表現を用いて表すか，ファンクションポイント法などを用いて一つの評価値にまとめる．

3. 対応言語の増加

現在，メトリクスを表示できるのはJava言語のみである．このため，Java以外のファイルは一律で黒色になってしまい，違いが分かりづらい．先行研究では拡張子に応じてグラフィックを変更していた．この機能を再実装した上でメトリクスを反映する事で，より多くの情報をユーザーに提示できる．

4. リポジトリ内探索の動機付け

リポジトリ内探索の動機付けのため，ゲーミフィケーションを用いる．ゲーミフィケーションとは，あるタスクにゲームの要素を持ち込むことでユーザーの動機付けを行うものである[11]．リポジトリ内探索をゲーム化することでプログラミング知識のないユーザーが積極的にリポジトリ探索を行いやすい環境を作る．前節の目標の達成のため，過度に複雑な部分は難易度を上げ，ソースコードの複雑さがゲームプレイに大きく影響を及ぼす実装が望ましい．

5. パッケージ化

サーバーサイド側のソースコードと共に本ウェブアプリケーションをパッケージ化し，Web上で公開する．これによって，多くのユーザーが気軽に本ウェブアプリケーションを利用してもらうことができる．

6. 関連研究

本研究を行うにあたり，実装の参考となった研究について解説する．

6.1 CodeHouse

CodeHouse は，VR を用いたソフトウェア可視化ツールである [12]．シリンダー状のオブジェクト内にモジュールを表す立方体オブジェクトを配置する事によって，シリンダー中央にいるプレイヤーはソフトウェアのモジュールを一望することが出来る．これは囚人監視システムとして有名なパノプティコンをヒントに，ソフトウェアのモジュールをどこに配置しても確認できることを意識したものである．

6.2 CodeCity

CodeCity は，クラスを建物に，パッケージを地区に見立て，ソフトウェアを一つの都市として表現するソフトウェア可視化ツールである [13]．元来複雑であるソフトウェア構造の過度な単純化を避け，ソフトウェア構造を徐々に理解する様子を大規模な都市を徐々に探索していくものとして実装しているのが特徴である．

6.3 IslandViz

IslandVis では，全てのモジュールが島として表現されている [14]．パッケージは島の地区，クラスは建物に該当する．こちらも CodeHouse と同様に VR を用いてソフトウェア構造を可視化するが，CodeHouse が 360 度全方位を見渡すような作りなのに対して，IslandVis は俯瞰視点を用いている．

7. 結言

本研究では，リポジトリからロールプレイングゲーム風 2 次元マップを生成するアプリケーション「GitQuest」に対して，リポジトリ内のソースコードメトリクスを取得，GitQuest のグラフィックに反映させる形で可視化を行った．

具体的に，ソースコードメトリクスをサーバー上の WebAPI から取得，LOC，FOC を用いて導いたコード規模を GitQuest 上のキャラクターグラフィックに色として反映させた．また，マップに対してファイルの存在を示唆するための建物オブジェクトの配置，ディレクトリ深度によるマップグラフィックの変更，同一ディレクトリに対するマップの固定化などを行った。

これによって，ユーザーにソースコードの規模を視覚的に印象付けることができるようになった．これをさらに発展させる事によって，開発者以外のユーザーによるリポジトリ内探索の補助が可能となり，ソフトウェア工学的知見の収集に役立つと考えられる．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系 水野修教授，崔恩瀞助教に厚く御礼申し上げます．本報告書執筆にあたり執筆，校閲等全面的に協力頂きました本学設計工学専攻 近藤将成先輩，西浦生成先輩，情報工学専攻 國領正真先輩，塩津拓真先輩，実装にあたって API の製作に協力頂いた，本学情報工学専攻 舟山優先輩，全般にわたって貴重な助言を多数いただきました情報工学課程 山本凱君，杉浦智基君をはじめとする，ソフトウェア工学研究室の皆さん，学生生活を通じて著者の支えとなった家族や友人に深く感謝致します．

参考文献

[1] M. Lanza, “The evolution matrix: Recovering software evolution using software visualization techniques,” Proceedings of the 4th international workshop on principles of software evolution, pp.37–42, 2001.

[2] Git–everything-is-local, Git（オンライン）, 入手先〈https://git-scm.com/〉（参照 2020-2-10）.

[3] 井口貴紀，“大学生のゲーム利用実態,” 情報通信学会誌，vol.33，no.2，pp.41–51，2015.

[4] Facebook, Inc., React A JavaScript library for building user interfaces, React（オンライン）, 入手先〈https://reactjs.org/〉（参照 2020-2-10）.

[5] GitHub, Inc., GitHub API v3 — GitHub Developer Guide, GitHub Developer（オンライン）, 入手先〈https://developer.github.com/v3/〉（参照 2020-2-7）.

[6] gskinner, CreateJS A suite of modular libraries and tools which work together or independently to enable rich interactive content on open web technologies via HTML5, CreateJS（オンライン）, 入手先〈https://createjs.com/〉（参照 2020-2-10）.

[7] W.W.W. Consortium, 4.12.4. The canvas element, HTML 5.3（オンライン）, 入手先〈https://www.w3.org/TR/html53/semantics-scripting.html#the-canvas-element〉（参照 2020-2-10）.

[8] S.H. Kan, Metrics and models in software quality engineering, Addison-Wesley Longman Publishing Co., Inc., 2002.

[9] D. Bau, davidbau/seedrandom, GitHub（オンライン）, 入手先〈https://github.com/davidbau/seedrandom〉（参照 2020-2-10）.

[10] D. Bau, Welcome to Python.org, Python（オンライン）, 入手先〈https://www.python.org/〉（参照 2020-2-10）.

[11] S. Deterding, “Gamification: designing for motivation,” Interactions, vol.19, no.4, pp.14–17, 2012.

[12] A. Hori, M. Kawakami, and M. Ichii, “Codehouse: Vr code visualization tool,” 2019 Working Conference on Software Visualization (VISSOFT), pp.83–87, IEEE, 2019.

[13] R. Wettel and M. Lanza, “Codecity: 3d visualization of large-scale software,” Companion of the 30th international conference on Software engineering, pp.921–922, 2008.

[14] M. Misiak, A. Schreiber, A. Fuhrmann, S. Zur, D. Seider, and L. Nafeie, “Islandviz: a tool for visualizing modular software systems in virtual reality,” 2018 IEEE Working Conference on Software Visualization (VISSOFT), pp.112–116, IEEE, 2018.