SuperSQL+Unityによるデータの3Dビジュアライゼーション

藤本　樹† 五嶋　研人† 遠山　元道†

† 慶應義塾大学 理工学部 情報工学科 〒223–8522 神奈川県横浜市港区日吉 3–14–1
E-mail: †{tatsu,goto}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし 様々なデータを各々の分野で扱う際に分析、もしくはそのデータを利用し何かを相手に伝える際、図や表、グラフといった方法でビジュアライズすることが求められる．より直感的に理解しやすい方法として 3 D でのビジュアライズの手法を提案する．3D は 2D より多くの情報を一度に示すことができる．しかし、3D ビジュアライズを実現するためには様々な知識や労力が必要とされる．そのため、今回 SuperSQL と呼ばれる SQL の拡張言語を用いる．SuperSQL はクエリの記述によりデータベースから取得したデータにユーザの指定する形のレイアウトを施す、様々な種類の構文化文書を作成する．この SuperSQL を拡張し、Unity と組み合わせることで宣言的なクエリ記述でユーザビリティの高い 3 D ビジュアライズの実現を提案する．

キーワード SQL, SuperSQL, Unity, データビジュアライズ

1. はじめに

様々な種類のデータを各々の分野で扱う方々がそのデータを見て何か行動を起こす、そのデータを使って相手に結果や考えを伝えるなどの際に、図や表、グラフといった方法でデータビジュアライズをすることが求められる．今回その手法の一つとして 3D ビジュアライズを提案する．3 D でデータビジュアライズすることにより、2D よりも多くの情報を一度にビジュアライズすることができ、動きなどを用いることでより直感的に理解できる形でビジュアライズすることが可能である．しかし、3D ビジュアライズの実現のためには、D3.js や Unity などのプログラミングの知識や労力が必要とされる．そこで、本論文では SuperSQL という SQL の拡張言語を用いることで、宣言的に 3D ビジュアライズの実現を提案する．

SuperSQL とは、クエリを記述することで関係データベースの出力結果を構造化し、多様なレイアウト表現を可能とする SQL の拡張言語である．通常の SQL ではフラットな一次元の表のみの作成が可能だが、SuperSQL を用いることにより様々なレイアウトの表を作成することができる．また HTML、PHP、PDF などを生成することもでき、これらは実際よりもはるかに少ないコードで記述することができる．この SuperSQL を用いることで関係データベース内のデータをビジュアライズするための 3D 空間の作成を実現した．

以下、本論文の構成を示す．第 2 章では関連研究と関連技術について述べ、第 3 章では SuperSQL の概要を説明し、第 4 章では 3D 空間でのデータビジュアライズについて述べ、第 5 章では評価について述べ、第 6 章では結論と今後の課題について述べる．

2. 関連研究・技術

2.1 Unity

本システムではレンダリングのためのツールとして Unity を採用している．Unity はゲーム開発用ツールであるが、現在はゲームのみならずバーチャルリアリティ技術にも応用されている．また、ユーザが作成した 3D オブジェクトやスクリプトなどを公開し、配布、販売するためのプラットフォームも存在し、これにより豊富な拡張機能を実現でき、よりクオリティの高い開発を行うことができる．

2.2 宣言的 3D ビジュアライズ

Stefan Lemme らは HTML に様々な核となるエレメントを作成し、ウェブ上で宣言的に 3D シーンを作成することを実現した．このシステムは拡張性やユーザビリティの高いシステムであると述べられている [5].

2.3 インタラクティブシステム

Arvind Satyanarayan らは UI ツールキットを UI のツールキットとリアクティブなプログラミングを用いてデータビジュアライゼーションのための宣言的インタラクションデザインを提案した [6]. Eugene Wu らはインタラクティブなビジュアライゼーションを表現するための SQL ライクな言語である、DeVIL 言語を開発した．DeVIL 言語を用いることで宣言的に様々なデータビジュアライズのためのインタラクティブシステムを作成することが可能であると述べられている [7]. Mark Simpson らは VR 技術を用い、3D 散布図を知覚する際にコントローラを用いて散布図を回転させるよりもユーザ自身が移動して知覚するほうがユーザの能力に左右されにくい結果となることを示した [8]. 本研究では VR 技術を用いて、可視化したデータ内を歩き回ることが可能である．現在コントローラを用いた操作は移動のみとなっているが、インタラクティブなシステムを VR 空間内に実現することで、より幅の広いデータビジュアライゼーションを実現することが可能である．

3. SuperSQL とは

SuperSQL は関係データベースの出力結果を構造化し、多様なレイアウト表現を可能とする SQL の拡張言語であり、慶應義塾大学遠山研究室で開発されている [1] [2]. そのクエリは SQL の SELECT 句を GENERATE <*media*> <*TFE*> の

構文を持つ GENERATE 句で置き換えたものである．ここで <*media*> は出力媒体を示し、HTML、PDF などの指定ができる．また <*TFE*> はターゲットリストの拡張である Target Form Expression を表し、結合子、反復子などのレイアウト指定演算子を持つ一種の式である．

3.1 アーキテクチャ

SuperSQL にはクエリを受け取り SQL クエリとレイアウト式に分ける Parser と SQL クエリによるデータベースからの取得結果と表の構造情報を受け取りデータ整形を行う DataConstructor、構造化されたデータを受け取りファイルを生成する CodeGenerator とに大きく分かれる．本論文では CodeGenerator 部分を拡張することで SSQL クエリを Parser と DataConstructor で解析し、C#ファイルと XML ファイルを出力するというアーキテクチャになっている．

図 1 に本論文における SuperSQL 処理系のアーキテクチャを示す．

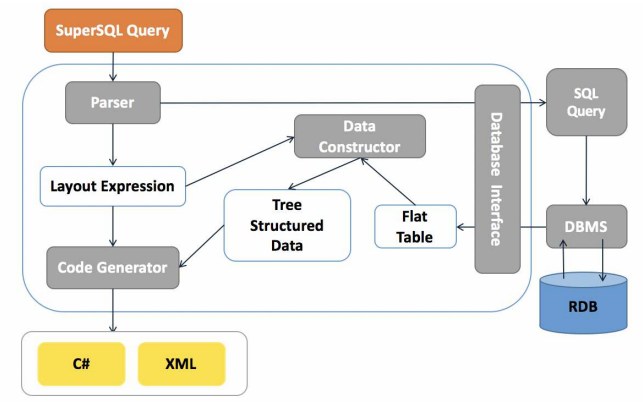


図 1　SuperSQL 処理系のアーキテクチャ

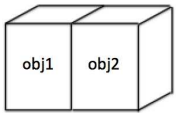
3.2 結　合　子

結合子はデータベースから得られたデータをどの方向 (次元) に結合するかを指定する演算子であり、以下の 3 種類がある．括弧内はクエリ中の演算子を示している．

- 水平結合子 (,)

データベースから得られたデータに沿って、3D オブジェクトを x 軸方向に配置する．

例：obj1, obj2



- 垂直結合子 (!)

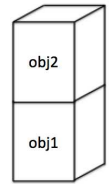
データベースから得られたデータに沿って、3D オブジェクトを y 軸方向に配置する．

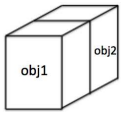
例：obj1! obj2

- 深度結合子 (%)

データベースから得られたデータに沿って、3D オブジェクトを z 軸方向に配置する．

例：obj1 % obj2





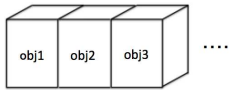
3.3 反　復　子

反復子は指定する方向に、データベースの値があるだけ繰り返して表示する．以下、その種類について述べる．

- 水平反復子 ([ ],)

データインスタンスがある限り、その属性のデータの 3D オブジェクトを x 軸方向に繰り返し配置する．

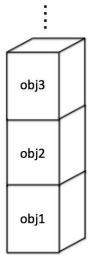
例：[Obj],



- 垂直反復子 ([ ]!)

データインスタンスがある限り、その属性のデータの 3D オブジェクトを y 軸方向に繰り返し配置する．

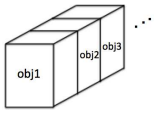
例：[Obj]!



- 深度反復子 ([ ]%)

データインスタンスがある限り、その属性のデータの 3D オブジェクトを z 軸方向に繰り返し配置する．

例：[Obj]%



3.4 複合反復子

ある方向 (次元) での反復の上限を指定し、これを超える場合に次に指定した方向 (次元) に伸長する．

[<TFE>] < 結合子 >< 回数 >< 結合子 >< 回数 >...< 結合子 >

例：[Obj],3!
3D オブジェクトを x 軸方向に 3 つ並べたものを、y 軸方向へ伸長していく.

例：[Obj],4%3!
3D オブジェクトを x 軸方向に 4 つ並べたものを、z 軸方向に 3 つ並べていき、まだ出力するオブジェクトがある場合はこの動作を y 軸方向へと伸長していく.

3.5 関　　数

SuperSQL にはいくつかの関数が存在する. 本研究ではこの関数をメインに実装を行った. その詳細は次章で説明する.

4. 3D 空間でのデータビジュアライズ

この章では 3D 空間でのデータビジュアライズについて説明し、実際にこのシステムを利用する際のクエリとその生成結果について述べていく.

4.1 システム概要

まずはじめにデータビジュアライズをしたいデータをデータベースに格納しておく. SuperSQL のクエリを実行すると、そのデータに基づいて XML ファイルと C#ファイルが生成される. XML ファイルにはオブジェクトの情報や付与されている要素、レイアウトが記述されている. C#ファイルはその XML ファイルを読み込みオブジェクトを生成し、色やアニメーションを付与、またレイアウトの情報から各オブジェクトの位置決定を行っている. この 2 つのスクリプトファイルを Unity にインポートし、実行することでクエリで指定したデータビジュアライズが実現される. またアセットを用いる場合にはアセットも同時に Unity にインポートする必要がある. この過程を図 2 に示す.

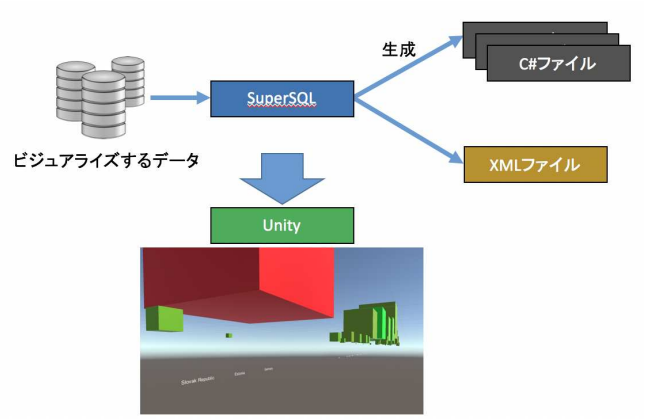


図 2　3D 空間でのデータビジュアライズ概要

4.2 関　　数

本研究では SuperSQL の関数を用いてデータビジュアライズを実現した. 関数は主にオブジェクト生成関数と、オブジェクトの表示属性を指定する関数とに分けられる. それぞれの一般例を示す. -オブジェクト生成関数の一般例

・object(object_name, para1, …, paraN)
・asset(asset_name, size)
object_name: cube, cuboid, sphere, torus, pyramid
asset_name: ユーザーが作成したオブジェクト名

-オブジェクトの表示属性を指定する関数の一般例

att_func1(att_func2(…att_funcN(オブジェクト生成関数, …), …)
att_func: location 関数, scale 関数, rotate 関数, hop 関数, pulse 関数, color 関数, color_gradient 関数
att_func1~att_funcN は省略可

4.2.1 オブジェクト生成関数

- object 関数

object 関数は生成するオブジェクトの形とオブジェクト毎に必要なパラメータを指定して、オブジェクトを生成する. この関数を使ってユーザーは cube、sphere、torus、cuboid、pyramid を生成することができる. 下記にオブジェクト (torus) を生成する際の object 関数の記述例とその生成結果を示す.
-object 関数の記述例
object(”torus”, r1, r2)

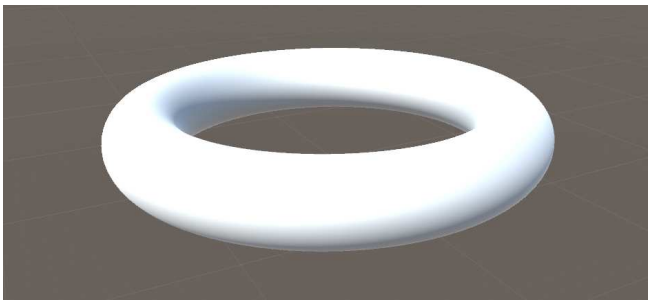


図 3　オブジェクト (torus) の生成例

- asset 関数

asset 関数は object 関数とほとんど同じだが、アセットと呼ばれるユーザーにより既に作成されたオブジェクトを指定し、そのサイズを共に与えることでオブジェクトを生成する. 下記にアセット (Windmill) を生成する際の asset 関数の記述例とその生成結果を示す.
-asset 関数の記述例
asset(”asset 名”, size)

4.2.2 オブジェクトの表示属性を指定する関数

- location 関数 location 関数はオブジェクトの絶対位置を指定する関数である. x、y、z 座標をパラメータとして与えることでオブジェクトの位置を指定する.

-location 関数の記述例
location(… , x 座標, y 座標, z 座標)

- scale 関数 scale 関数はオブジェクトの大きさを指定する関数である. size をパラメータとして与えることでオブジェクトの大きさを指定する.

図 4　アセット (Windmill) の生成例

-scale 関数の記述例
scale(…, サイズ)

- color 関数 color 関数はオブジェクトに対して、色を文字列で記述して付与する．使用できる色には”red”, ”blue”, ”green”, ”black”, ”clear”, ”cyan”, ”gray”, ”grey”, ”yellow”, ”white”, ”magenta”が存在する．下記に color 関数の記述例を示す．

-color 関数の記述例
color(…, ”color”)

- color_gradient 関数

color_gradient 関数はオブジェクトに対して、数値を色に変換して付与する．この際, 値の最小値と最大値も同時に記述する必要がある．そこで、SuperSQL の集約関数である min 関数と max 関数を用いる．下記に color_gradien 関数の記述例と生成例を示す．

-color_gradient 関数の記述例
color_gradient(… , ”最小値の時の色の名前”, min(属性名), ”最大値の時の色の名前”, max(属性名), 属性名)

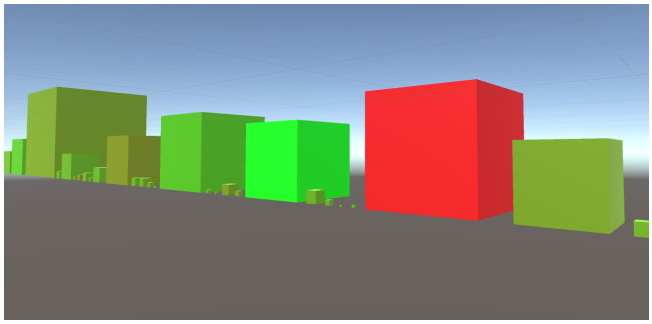


図 5　color_gradient の例

また、オブジェクトの表示属性を指定する関数にはアニメーションに関する関数も存在する．これには hop 関数、pulse 関数、rotate 関数が属する．ここではその一例として pulse 関数の記述例とイメージを図 6 に示す．

- pulse 関数

pulse 関数は大きさの倍率 (1 が等倍) と速度 (1 が等倍) を記述することで、オブジェクトを伸縮させることができる．

-pulse 関数の記述例
pulse(…, 倍率, 速さ)

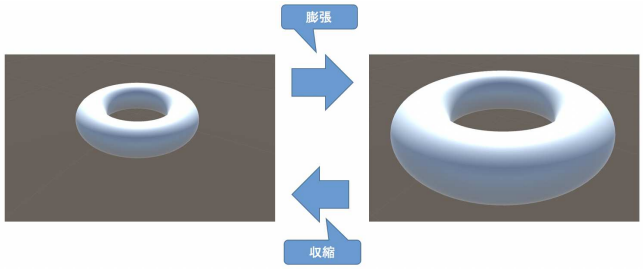


図 6　pulse の例

4.3　関数の入れ子

今までに説明してきた関数を入れ子の形で用いることにより、一つのオブジェクトに対して複数の要素を付与することができる．この際、object 関数と asset 関数は子を持つことができないが、その他の関数は順序に関係なく子を持つことができる．また、同じ関数が繰り返された際にはその要素を付与するオブジェクトにより近い側の関数が優先される．その記述例は以下のようになる．

```
color(
        hop(
                object(”object1”, …),
                速さ, 頂点, ”軸”
        ), color(
                object(”object2”, …),
                ”色 2”
        ), ”色 1”
)
```

この記述例で object1 には hop の情報と色 1 の情報が付与されるが、object2 には色 1 の情報は付与されず、色 2 の情報のみが付与される．

4.4　データビジュアライズ例

以下に 3D 空間でのデータビジュアライズの例として質問文とその結果を示す．データベースで今回は countries テーブル、infomation テーブル、regions テーブルの 5 つのテーブルを用いる．- countries テーブル (属性:id, name, region) name は国の名前、region は regions テーブルへの外部キーを表す．
- information テーブル (属性:c_id, year, pop, cab, gdp)
c_id は countries テーブルへの外部キー、year は年度、pop は人口、cab は経常収支、gdp は国内総生産を表す．
- regions テーブル (属性:id, name)
name は州の名前を表す．

- 例 1

```
GENERATE Unity_dv
[pulse(
        color_gradient(
                object(”sphere”, i.pop),
                ”blue”, min[i.cab], ”red”, max[i.cab],
                i.cab
        ), 1.2, i.gdp
)],
from countries c, information i, regions r
where r.name=”Europe” and c.region=r.id and i.c_id=c.id
and i.year=2017
```

上記の質問文は 2017 年度のヨーロッパ州の国々における人口、経常収支、国内総生産をそれぞれ色やアニメーションなどの要素にマッピングしてデータビジュアライズする例である.

まず 3 行目にあるオブジェクトの球のサイズに各国毎の人口がマッピングされる. その後、その球に対して 1 行目、2 行目、の pulse 要素や color_gradient 要素が付与されていく. 4 行目で球の色は経常収支がヨーロッパ州の国々の中で一番小さいものが青色になり、一番大きいものは赤色になる. それ以外の国々は値が小さいものほど青色に近く、値が大きいものほど赤色に近い色になり、中間の色は紫色になる.

そして 5 行目で pulse の速さに gdp をマッピングしている. よって pulse がより早く行われているものほど gdp の値が大きいということが判別できるようになっている. その生成結果を図 7 に示す.

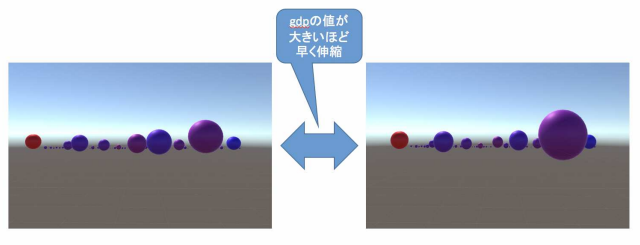


図 7　例 1 の生成結果

また、先ほどの例では pulse に経常収支を割り当てたが、ビジュアライズした際に分かり辛いと思った際には pulse を hop に変えて一番高く跳ね上がるものほど gdp の値が大きいという風にビジュアライズし直すことも少しのクエリの変更で可能である.

以下に pulse を hop に変更した際の質問文とその生成結果を示す.

- 例 2

```
GENERATE Unity_dv
[hop(
        color_gradient(
                object(”sphere”, i.pop),
                ”blue”, min[i.cab], ”red”, max[i.cab],
                i.cab
        ), 1, i.gdp, ”y”
)],
from countries c, information i, regions r
where r.name=”Europe” and c.region=r.id and i.c_id=c.id
and i.year=2017
```

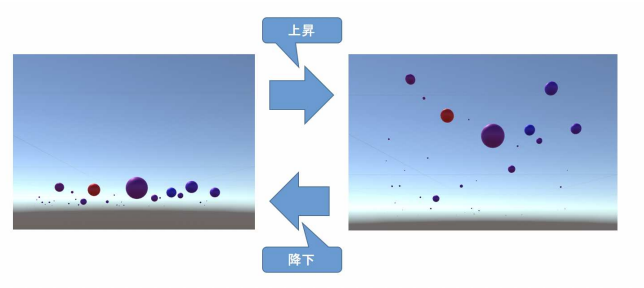


図 8　例 2 の生成結果

4.5　ヘッドマウントディスプレイ (HMD)

本研究では HMD を用いることを想定している. HMD を用いて 3D 空間に没入することでより人の感性を刺激し、可視化したデータの情報を強く印象づけることができる.

5.　評　　価

本研究の評価として本システムにおけるそれぞれの要素に対してマッピング可能なデータの次元数を示す.

- 各オブジェクト、関数にマッピング可能なデータの次元数

オブジェクト名	データの次元
cube	1
cuboid	3
sphere	1
torus	2
pyramid	2

表 1　各オブジェクトにマッピング可能なデータ次元数

関数名	データの次元
color_gradient	1
hop	2
pulse	2
rotate	3
location	3

表 2　各関数にマッピング可能なデータ次元数

表 1、表 2 は各オブジェクト毎にマッピング可能なデータの次元数と各関数毎にマッピング可能なデータの次元数を示したものである．同時に可視化可能なデータの次元数は単一のオブジェクトに対して複数の関数を使用できるので、最大で 14 種類のデータをマッピングすることができる．しかし、人間の識別能力に依存する部分が大きいため、レイアウトはデータの次元数を増やすほど複雑になっていく．

6. ま と め

6.1 結　　論

本研究では SuperSQL を用いて多次元データを 3D ビジュアライズするシステムを提案した．このシステムにより、SuperSQL のクエリからデータベースに格納されたデータを様々なレイアウトで 3D ビジュアライズすることが可能となった．宣言的にクエリを記述することができるため、ユーザーがクエリを記述する段階でビジュアライズのイメージがしやすい．また，比較的短いクエリで実行が可能なためレイアウトを変更する際に少ない労力で行うことができる．これらの点で本システムは有用である．

6.2 今後の課題

本システムにおいてコントローラーを使用した操作は移動のみであるが、より効果的なビジュアライズを実現するためにインタラクティブなシステムを開発し使いやすいシステムにしていく必要がある．そこで、オブジェクトにリンクを付与し、コントローラの操作によってその情報に関連したシーンを閲覧することを可能にするなどのことをこれから実現していく．

また, 現在のシステムでは利用できるアニメーションの種類が少なく、様々なビジュアライズを試して最適な方法を見つけるにはまだ物足りない．そのため、より有効なビジュアライズを実現するためのアニメーションを増やしていくことは重要である．しかし、人間の識別能力には限界があり、ある要素がどの範囲での変化であればその変化を識別可能であるかといったことも考える必要がある．それらを考慮しアニメーションを増やしていくことでさらに多次元のデータをビジュアライズすることが大きな課題である．

文　　献

[1] SuperSQL: http://SuperSQL.db.ics.keio.ac.jp

[2] M. Toyama: ”SuperSQL: An Extended SQL for Database Publishing and Presentation”，Proceedings of ACM SIGMOD ’98 International Conference on Management of Data，pp. 584-586，1998

[3] Unity マニュアル: https://docs.unity3d.com

[4] 中西 厚友, 遠山 元道. ”SuperSQL による仮想 3 次元空間の自動生成”, 慶應義塾大学, 2008.

[5] Stefan Lemme, JanSutter, Christian Schlinkmann, Philipp Slusallek. ”The Basic Building Blocks of Declarative 3D on the Web”, Web3D, pp.17-25,2016.

[6] Arvind Satyanarayan, Kanit Wongsuphasawat, Jeffrey Heer. ”Declarative Interaction Design for Data Visualization”, Proceedings of the 2017th annual ACM symposium on User interface software and technology, pp.669-678, 2014.

[7] Eugene Wu, Fotis Psallidas, Zhengjie Miao, Haoci Zhang, Laura Rettig, Yifan Wu, Thibault Sellam. ”Combining Design and Performance in a Data Visualization Management System”, Published at a Creative Commons Attribution License, CIDR, 2017.

[8] Mark Simpson, Jiayan Zhao, Alexander Klippel. ”Take a Walk:Evaluating Movement Types for Data Visualization in Immersive Virtual Reality”, Published at Immersive2017, workshop of IEEE VIS 2017.