react cop2

ono

2024 年 1 月 5 日

目 次

1 はじめに

1.1 研究背景

1.2 研究課題

1.3 研究目的

1.4 本論文の構成

2 関連研究

2.1 COP の概要

2.2 React COP

2.3 EventCJ に複合層を導入

3 提案手法

本研究では、関連研究にある複合層、多層の機能追加 react cop の改善点を洗い出し、それを解決するための機能を追加した。本章では、本研究で提案する手法について述べる。

react cop では、レイヤーの活性化情報を useState を用いて管理している。state 変数はクラスのインスタンスを用いている。しかし、state 変数は参照型のため、state 変数の値を変更しても再レンダリングが行われない。そのため、レイヤーの活性化情報を更新しても、再レンダリングが行われない。これを解決するために、state 変数を参照型のクラスのインスタンスから、イミュータブルなデータに変更する必要がある。この変更は react cop を大きく刷新することと同等であるため、react cop2 として新しく実装することとした。

react cop では、レイヤーのパラメーターと layer の活性化情報を別で管理している。そのため、レイヤーの活性化とレイヤーのパラメーターで 2 度レイヤー名を指定する必要がある。これは、レイヤーの管理が煩雑になる原因となっている。この問題を解決するために、レイヤーを一つのオブジェクトとして管理することとした。

以降では、react cop2 の実装について述べる。ReactCOP で代替となる機能を提供するために、react cop2 では、以下の機能を提供する。ReactCOP2 では、カスタムコンテキストを用いて、レイヤーの操作や取得などの機能を提供する。react cop から改善された機能を以下に示す。

3.1 代替する機能

(1) レイヤーパラムの設定・取得

React cop では、レイヤーパラムの設定・取得ができる useLayerParams というカスタムフックを提供している。このカスタムフックは、以下のように使用する。

```
1 // レイヤーパラム
2 const [getHoge, setHoge] = useLayerParams('hoge', ["Hoge"]);
3 getHoge() // hoge
4 setHoge('fuga',"Hoge")
5 getHoge() // fuga
```

useLayerParams の第一引数には、レイヤーパラムの初期値、第二引数には、レイヤー名を指定することで、レイヤーパラムの値を設定できる。レイヤー名は複数指定することができる。setHoge の第一引数には、レイヤーパラムの値、第二引数には、レイヤー名を指定することでレイヤーパラムの値を設定できる。getHoge の引数には、レイヤー名を指定することで、レイヤーパラムの値を取得できる。引数にレイヤー名を指定しない場合は、活性化したレイヤーのレイヤーパラムの値を取得する。ただし、活性化したレイヤーが複数ある場合は、最初に取得したレイヤーのレイヤーパラムの値を取得する。レイヤーの並び順は、レイヤー名を登録した順番である。

ReactCOP2 では、レイヤーパラムの設定ができる setLayerParams、取得ができる getLayerParams というメソッドを提供する。このメソッドは、以下のように使用する。setLayerParams の第一引数には、レイヤー名、第二引数にはレイヤーパラムの値を指定することで、レイヤーパラムの値を設定できる。getLayerParams の引数には、レイヤー名を指定することで、レイヤーパラムの値を取得できる。

ReactCOP2 では、レイヤーパラムの設定ができる

(2) レイヤーの活性化・非活性化

ReactCOP では、レイヤーの活性化・非活性化ができる useLayerManager というカスタムフックを提供している。このカスタムフックは、以下のように使用する。

```
1 const layerManager = useLayerManager();
2 layerManager.activateLayer("Float");
3 layerManager.deactivateLayer("Integer");
```

activateLayer の引数には、レイヤー名を指定することで、レイヤーを活性化できる。deactivateLayer の引数には、レイヤー名を指定することで、レイヤーを非活性化できる。

(3) レイヤーの活性化情報の取得

ReactCOP では、レイヤーの活性化情報を取得できる useLayerManager というカスタムフックは getLayerState というメソッドを提供している。このメソッドは、以下のように使用する。

```
1 const layerManager = useLayerManager();
2 const layerState = layerManager.getLayerState();
3
4 layerState.Float
5 layerState.Integer
```

layerState. レイヤー名で、レイヤーの活性化情報を取得できる。ただし、あらかじめレイヤーを活性化・非活性化していないと、レイヤーの活性化情報は取得でず、undefined が返される。

(4) レイヤーが活性化しているかどうかの判定

ReactCOP では、レイヤーが活性化しているかどうかの判定ができる useLayerManager というカスタムフックは isActiveLayer というメソッドを提供している。このメソッドは、以下のように使用する。

```
1 const layerManager = useLayerManager();
2 // レイヤーがactive かどうかを判定
3 layerManager.isActiveLayer("Float")// true or false
```

isActiveLayer の引数には、レイヤー名を指定することで、レイヤーが活性化しているかどうかを判定できる。レイヤーが活性化している場合は、true が返される。

3.2 改善点

(1) 改善点一覧

- layer の de/active 時に新しいレイヤーを定義できないようにする react cop では、layer の de/active 時に新しいレイヤーを定義できてしまう。そのため、意図しないレイヤーが簡単に定義できてしまう。またレイヤーの管理が煩雑になる。コレを解決するために、layer の de/active 時に新しいレイヤーを定義できないようにする
- layer params は in/active の両方の状態を持つ
- layer params は layer の in/active 状態に依存をするようにしたい
- layer params の値を入れるときに新しい layer を定義できないようにする
- layer grop 的なのをついか
- layer の活性化条件を定義できる
  - 複合層
  - 多層
- layer の活性化は排反
- typescript での実装
- テストの追加

(2) layer の de/active 時に新しいレイヤーを定義できないようにする

ReactCOP では、layer の de/active 時に新しいレイヤーを定義できてしまう。layer の de/active 時に新しいレイヤーを定義できると意図しないレイヤーが簡単に定義できてしまう。またレイヤーの管理が煩雑になる。

ReactCOP2 では、layer の de/active 時に新しいレイヤーを定義できないようにする。これによって、意図しないレイヤーが簡単に定義できなくなり、レイヤーの管理が煩雑にならない。

(3) layer params は in/active の両方の状態を持つ

(4) layer params は layer の in/active 状態に依存をするようにしたい

(5) layer params の値を入れるときに新しい layer を定義できないようにする

(6) layer grop 的なのをついか

(7) layer の活性化条件を定義できる

(8) layer の活性化は排反

ソースコード 1: hoge

```
1 // このとき、Float と Integer のレイヤの活性化は排反でよい気がする
2 const [getHoge, setHoge] = useLayerParams('', ["Float", "Integer"]);
3
4 // いちいち切り替えがめんどう
5 layerManager.deactivateLayer("Integer");
6 layerManager.activateLayer("Float");
```

```
7
8
9  // 排反ではないから２つ条件含むのどうなん?
10 <Layer condition={layerState.Float && !layerState.Integer}>
```

3.3 実装する内容

(1) typescript での実装

(2) テストの追加

3.4 評価方法

- 実装前と後で、できることの違いを比較する。

4 実装

4.1 layer の de/active 時に新しいレイヤーを定義できないようにする

コードは以下のようになる。レイヤーの名前が存在するかどうかを確認し、存在しない場合はエラーを出すようにしている。

4.2 ts の導入

4.3 テストの導入

4.4 各種、具体的な実装内容

5 評価

本章では，提案手法の評価を行う．

5.1 できるようになったこと

6 まとめ

7 参考文献