react cop2

ono

2024 年 1 月 6 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。

(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。

ReactCOP2 では、レイヤーの活性化・非活性化ができる activateLayer、inactivateLayer 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。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。inactivateLayer 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。

ReactCOP2 では、レイヤーの活性化情報を取得できる getLayer 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。getLayer の引数には、レイヤー名を指定することで、レイヤーの情報を取得できる。

(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。

```
const layerManager = useLayerManager();
// レイヤーがactive かどうかを判定
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。

ReactCOP2 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メソッドは提供しない。代わりに、レイヤーの活性化情報を取得できる getLayer メソッドから、レイヤーが活性化情報を取得すれば、レイヤーが活性化しているかどうかの判定ができる。TODO: ここにコードを書く

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 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
- 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 の活性化条件を定義できる

3.3 機能追加

(1) 機能追加一覧

- アクティブなレイヤーを取得する getActiveLayers: () =¿ Layers;
- レイヤーのパラメーターを取得する getLayersParam: (names: string[]) =¿ any[]; getAllLayersParams: () =¿ any; getActiveLayersParams: () =¿ any; getLayerParamsByGroup: (group: string) =¿ any;
- グループを指定してレイヤーを取得する getLayersByGroup: (group: string) =¿ Layers; getActiveLayersByGroup: (group: string) =¿ Layers; getInactiveLayersByGroup: (group: string) =¿ Layers;
- レイヤーを追加する addLayer: (layer: Layer) =¿ void;
- レイヤーを削除する removeLayer: (name: string) =¿ void;
- レイヤーの活性化を切り替える toggleLayer: (name: string) =¿ void;

- グループの設定をする setGroupConfig: (name: string, config: GroupConfig) =¿ void;
- レイヤーにグループを設定する setLayerGroup: (name: string — string[], group: string) =¿ void;
- レイヤーの活性化条件を設定する addDependency: (name: string, dependencies: DependencyGroup) =¿ void;

3.4 アクティブなレイヤーを取得する

ReactCOP でアクティブなレイヤーを取得するためには、レイヤーの活性化情報を取得し、活性化しているレイヤーを取得する必要がある。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、アクティブなレイヤーを取得する機能を提供することで、レイヤーの管理が簡単になる。

3.5 レイヤーのパラメーターを取得する

(1) 複数のレイヤーのパラメーターを取得する

ReactCOP では、複数のレイヤーのパラメーターを取得するためには、レイヤーのパラメーターを取得するためには、レイヤー名を指定してレイヤーのパラメーターを取得する必要がある。少し不便であるため、複数のレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

(2) 全てのレイヤーのパラメーターを取得する

ReactCOP では、全てのレイヤーのパラメーターを取得するためには、レイヤーのパラメーターを取得するためには、レイヤー名を指定してレイヤーのパラメーターを取得する必要がある。少し不便であるため、全てのレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

(3) アクティブなレイヤーのパラメーターを取得する

ReactCOP でアクティブなレイヤーのパラメーターを取得するためには、レイヤーが活性化しているかどうかの判定を行い、活性化しているレイヤーのパラメーターを取得する必要がある。これは、少し不便であるため、アクティブなレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

(4) グループを指定してレイヤーのパラメーターを取得する

RectCOP では、そもそもグループの概念がないため、グループを指定してレイヤーのパラメーターを取得する機能は提供していない。しかし、グループを指定してレイヤーのパラメーターを取得する機能を提供することで、レイヤーの管理が簡単になる。

3.6 レイヤーを追加する

ReactCOP では、明確にレイヤーを追加する機能を提供していない。レイヤーを活性化させるときやレイヤーのパラメーターを設定するときに、もしレイヤーが存在しない場合はレイヤーを追加するという処理を行っている。これは、いつレイヤーが追加されるかわからないため、レイヤーの管理が煩雑になる。

ReactCOP2 では、明確にレイヤーを追加する機能を提供することで、レイヤーの管理が簡単になる。

3.7　レイヤーを削除する

必要がないレイヤーを削除することで、レイヤーの管理が簡単になる。

3.8　レイヤーの活性化を切り替える

ReactCOP では、レイヤーの活性化を切り替える機能を提供していない。レイヤーの活性化を切り替えるには、レイヤーの活性化情報を取得し、活性化しているかどうかの判定を行い、活性化している場合は、レイヤーを非活性化し、非活性化している場合は、レイヤーを活性化する必要がある。これは、少し手間であるため、レイヤーの活性化を切り替える機能を提供することで、レイヤーの管理が簡単になる。

3.9　グループの設定をする

ReactCOP では、そもそもグループの概念がないため、グループの設定をする機能は提供していない。グループとは、レイヤーをグループ化することである。グループを設定することで、グループ内のレイヤー内で活性化するレイヤーを取得するなどの機能を提供することができる。

ReactCOP2 では、グループの設定をする機能を提供することで、レイヤーの管理が簡単になる。

3.10　レイヤーにグループを設定する

レイヤーがどのグループに属するか設定することができる。

3.11　レイヤーの活性化条件を設定する

レイヤーの活性化条件を設定することができる。レイヤーの活性化条件とは、レイヤーが活性化する条件である。例えば、レイヤー A とレイヤー B があるとき、レイヤー A が活性化しているときに、レイヤー B を活性化するという条件を設定することができる。条件の指定方法は、レイヤーの活性化状態の論理積、論理和を指定できることである。

(1)　Typescript での実装

ReactCOP では、Typescript での実装をしていない。Typescript での実装をすることで、コードの可読性と保守性が向上する。

(2)　テストの追加

ReactCOP では、テストをしていない。テストは、コードの品質を保つために必要である。

3.12　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