# Creating a ColorPicker Dialog

Military units usually have some kind of a logo or way to visually identify themselves. Right now, our fictional Battletech units only store a name and a faction affiliation. Let's add a field for a color as well.

If we're going to have a color field, we need some way to actually let the user pick the color. And *obviously*, if we're going to have a color picker, it should go in a modal dialog! :) (See previous comments about deliberate over-engineering.)

We'll need to build an input to show what the current color is and trigger the color picker dialog, as well as the dialog itself.

We've already got the ability to show a modal, so we only need to create a new modal component class that renders some kind of color picker component. We'll use the React-Color library, which provides a variety of color pickers in various styles.

```
yarn add react-color@2.13.8
```

> **Commit ba2b3ef: Add React-Color library**

The `ColorPickerDialog` itself will be very simple. We just need to track some kind of color value in its state, pass that to the color picker component when we render, and update the state when the user selects a different color. We'll also want to accept a callback prop that we can pass the new color to when the modal is closed successfully.

> **Commit 9c331f5: Add an initial ColorPickerDialog**

## common/components/ColorPickerDialog.jsx

```
import React, {Component} from "react";
import {connect} from "react-redux";
import {
    Modal,
    Button,
} from "semantic-ui-react";

import {SketchPicker} from "react-color";

import {closeModal} from "features/modals/modalActions";
import {noop} from "common/utils/clientUtils";

const actions = {closeModal};

export class ColorPickerDialog extends Component {
    constructor(props) {
        super();
        this.state = {
            color : props.color
        }
    }

    onSelectClicked = () => {
        this.props.colorSelected(this.state.color);

        this.props.closeModal();
    }

    onSelectedColorChanged = (colorEvent) => {
        this.setState({color : colorEvent.hex});
    }

    render() {
        const {closeModal} = this.props;

        return (
            <Modal
                closeIcon="close"
                open={true}
                onClose={closeModal}
                size="small"
            >
                <Modal.Header>Select Color</Modal.Header>
                <Modal.Content>
```

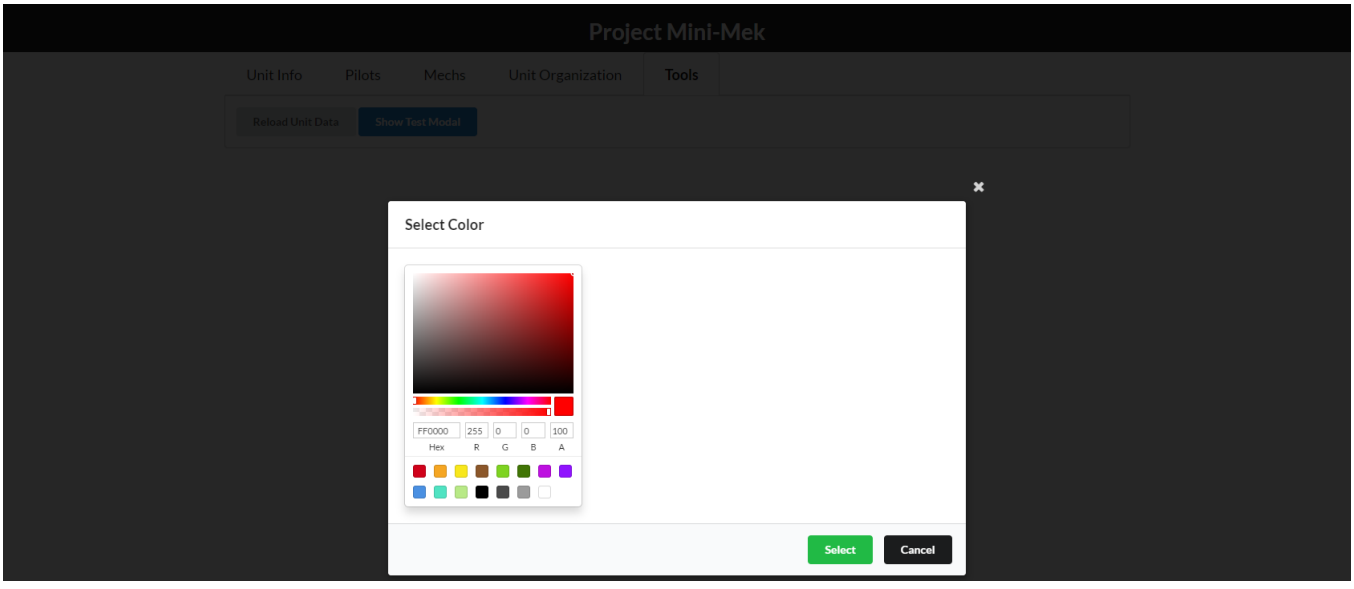
```
                    <SketchPicker
                        color={this.state.color}
                        onChangeComplete={this.onSelectedColorChanged}
                    />
                </Modal.Content>
                <Modal.Actions>
                    <Button positive onClick={this.onSelectClicked}>Select
</Button>
                    <Button secondary onClick={closeModal}>Cancel</Button>
                </Modal.Actions>
            </Modal>
        )
    }
}

ColorPickerDialog.defaultProps = {
    color : "red",
    colorSelected : noop
};

export default connect(null, actions)(ColorPickerDialog);
```

Pretty straightforward. We copy the initial color value from props to state in the constructor, and have all future changes apply to the component state. (We also added the `ColorPickerDialog` to the lookup table in `ModalManager` as well.)

Here's what it looks like if we show it:




(Yes, the fact that the modal is so much bigger than the color picker component annoys me, but the Semantic-UI modal layouts don't seem to have

much flexibility in size, and I don’t feel like messing with this.)