

```

/**
 * @file HudsonJunction.jsx
 * @author Joey Damico
 * @date September 25, 2019
 * @summary React JSX Component Class that is for Hudson Junction
Interlocking
 *
 * Extends the React Component Class and is the UI part of the Hudson
Junction Interlocking,
 * this class controls all the drawings of routes, and also gives a
visual representation
 * of that status of the interlocking
 */

// Import React Component
import React, { Component } from 'react';
// Import CSS style sheet
import '../css/Southern_Tier_Line/hudsonJunction.css';

// Import Images
// Switch Images
import SW_D_E from '../public/images/SW_D_E.png';
import SW_D_E_Lined from '../public/images/SW_D_E_Lined.png';
import SW_D_E_Occupied from '../public/images/
SW_D_E_Occupied.png';
import SW_D_E_R from '../public/images/SW_D_E_R.png';
import SW_D_E_R_Lined from '../public/images/
SW_D_E_R_Lined.png';
import SW_D_E_R_Occupied from '../public/images/
SW_D_E_R_Occupied.png';
import SW_U_W from '../public/images/SW_U_W.png';
import SW_U_W_Lined from '../public/images/SW_U_W_Lined.png';
import SW_U_W_Occupied from '../public/images/
SW_U_W_Occupied.png';
import SW_U_W_R from '../public/images/SW_U_W_R.png';
import SW_U_W_R_Lined from '../public/images/
SW_U_W_R_Lined.png';
import SW_U_W_R_Occupied from '../public/images/
SW_U_W_R_Occupied.png';

// Signal Images
import SIG_W from '../public/images/SIG_W.png';
import SIG_W_Clear from '../public/images/SIG_W_Clear.png';
import SIG_W_Stop from '../public/images/SIG_W_Stop.png';
import SIG_E from '../public/images/SIG_E.png';
import SIG_E_Clear from '../public/images/SIG_E_Clear.png';
import SIG_E_Stop from '../public/images/SIG_E_Stop.png';

// Color Constants For Drawing Routes
const Empty = '#999999';

```

```
const Green = '#75fa4c';
const Red = '#eb3323';
```

```
/**
 * The React JSX Component Class for the Hudson Junction Interlocking
 *
 * This class is a JSX React Component for the Hudson Junction
Interlocking, this will control all the UI for the comonent,
 * and the click events that will pass reference between the backend
and the user. This also controls drawing the
 * route drawings to show if a route(s) is setup in the interlocking
or if the route is occupied
 */
class HudsonJunction extends Component {
  /**
   * State
   * @summary Object that holds the state or status information for
the component
   *
   * This object holds all the information for the interlocking that
is required to display the routes
   * correctly
   *
   * Anything that has "this.props." is passed down from the CTC
interlocking class
   */
  state = {
    // Switch Status
    sw_1: this.props.status.sw_1,
    sw_3: this.props.status.sw_3,
    // Image File for the switch - Will change depending on route
    sw_1_src: SW_U_W,
    sw_3_src: SW_D_E,
    // Image File for the signals - Will change depending on route
    sig_2w_src: SIG_W,
    sig_2ws_src: SIG_W,
    sig_2e_src: SIG_E,
    sig_2es_src: SIG_E,
    // Colors for tail tracks - Will change depending on route
    tail_1_w: Empty,
    tail_2_w: Empty,
    tail_e: Empty,
    tail_nysw: Empty,
    // Information For Interlocking Routes
    occupied: this.props.status.occupied,
    routes: this.props.status.routes
  };
}

/**
```

```

    * componentWillReceiveProps()
    * @summary Function that updates the state of the component
    *
    * The data that is being changed is passed down from the CTC
classes in the simulation backend
    *
    * @param nextProps, the new data to set the component state too
    */
    componentWillReceiveProps(nextProps){
        this.setState({
            sw_1: nextProps.status.sw_1,
            sw_3: nextProps.status.sw_3,
            occupied: nextProps.status.occupied,
            routes: nextProps.status.routes
        });
    }
    // ---- END componentWillReceiveProps() ----

    /**
    * render()
    * @summary standard React function that draws the interlocking to
the screen
    */
    render() {
        // Clear all the drawings from the interlocking so if a train
clears the route is gone
        this.reset_drawings();
        // Set the switch images based off the state of each crossover
        this.set_switch_img();
        // Draw all the current routes in the interlocking
        this.set_route_drawings();

        // Returns the HTML to draw the interlocking and it's current
state to the screen
        return (
            <div>
                {/* Tags */}
                <div className="hudson_title">CP HUDSON JUNCTION</div>
                <div className="hudson_milepost">MP 63.4JS</div>
                {/* West Side Tail Tracks */}
                <div className="hudson_2_west" style={{background:
this.state.tail_2_w}}></div>
                <div className="hudson_1_west" style={{background:
this.state.tail_1_w}}></div>
                {/* Switches */}
                <div className="hudson_SW_3"
onClick={this.props.throw_sw_3}><img src={this.state.sw_3_src}/></div>
                <div className="hudson_SW_1"
onClick={this.props.throw_sw_1}><img src={this.state.sw_1_src}/></div>
                {/* East Side Tail Tracks */}
            </div>
        );
    }

```

```

        <div className="hudson_east" style={{background:
this.state.tail_e}}></div>
        <div className="hudson_nysw" style={{background:
this.state.tail_nysw}}></div>
        { /* Signals */ }
        <div className="hudson_sig_2es"
onClick={this.props.click_sig_2es}><img src={this.state.sig_2es_src}/
></div>
        <div className="hudson_sig_2e"
onClick={this.props.click_sig_2e}><img src={this.state.sig_2e_src}/></
div>
        <div className="hudson_sig_2w"
onClick={this.props.click_sig_2w}><img src={this.state.sig_2w_src}/></
div>
        <div className="hudson_sig_2ws"
onClick={this.props.click_sig_2ws}><img src={this.state.sig_2ws_src}/
></div>
    </div>
    );
}
// ---- END render() ----

/**
 * @summary Sets the drawing for the route through the
interlocking
 *
 * Function takes what routes are currently set in the
Interlocking class and displays that route in the UI, the drawing
 * will change depending on if the interlocking is occupied or not
 */
set_route_drawings() {
    // Setting the color of the tracks depending on if the
interlocking is occupied or not
    let color = null;
    if (this.state.occupied) {
        color = Red;
    }
    else {
        color = Green;
    }

    // Loop through all the routes
    for (let i = 0; i < this.state.routes.length; i++) {
        if (this.state.routes[i] === "W_1_1__|__1_hall_hudson" ||
this.state.routes[i] === "E_1_1__|__1_hudson_valley") {
            // Tail Tracks
            this.state.tail_1_w = color;
            this.state.tail_e = color;

            // The Route Is Occupied

```

```

    if (this.state.occupied) {
        // Switches
        this.state.sw_1_src = SW_U_W_Occupied;
        this.state.sw_3_src = SW_D_E_Occupied;

        // Signals
        this.state.sig_2w_src = SIG_W_Stop;
        this.state.sig_2ws_src = SIG_W_Stop;
        this.state.sig_2e_src = SIG_E_Stop;
        this.state.sig_2es_src = SIG_E_Stop;
    }
    // The Route Is NOT Occupied
    else {
        // Switches
        this.state.sw_1_src = SW_U_W_Lined;
        this.state.sw_3_src = SW_D_E_Lined;

        // Signals
        // West Bound Signals
        if (this.state.routes[i] === "W_1_1__|
__1_hall_hudson") {
            this.state.sig_2w_src = SIG_W_Clear;
            this.state.sig_2ws_src = SIG_W_Stop;
            this.state.sig_2e_src = SIG_E_Stop;
            this.state.sig_2es_src = SIG_E_Stop;
        }
        // East Bound Signals
        else {
            this.state.sig_2w_src = SIG_W_Stop;
            this.state.sig_2ws_src = SIG_W_Stop;
            this.state.sig_2e_src = SIG_E_Clear;
            this.state.sig_2es_src = SIG_E_Stop;
        }
    }
}
else if (this.state.routes[i] === "W_1_2__|
__2_hall_hudson" || this.state.routes[i] === "E_2_1__|
__1_hudson_valley") {
    // Set Tail Track Colors
    this.state.tail_2_w = color;
    this.state.tail_e = color;

    // The Route Is Occupied
    if (this.state.occupied) {
        // Switches
        this.state.sw_1_src = SW_U_W_R_Occupied;
        this.state.sw_3_src = SW_D_E_Occupied;

        // Signals
        this.state.sig_2w_src = SIG_W_Stop;

```

```

        this.state.sig_2ws_src = SIG_W_Stop;
        this.state.sig_2e_src = SIG_E_Stop;
        this.state.sig_2es_src = SIG_E_Stop;
    }
    // The Route Is NOT Occupied
    else {
        // Switches
        this.state.sw_1_src = SW_U_W_R_Lined;
        this.state.sw_3_src = SW_D_E_Lined;

        // Signals
        // West Bound Signals
        if (this.state.routes[i] === "W_1_2__|
__2_hall_hudson") {
            this.state.sig_2w_src = SIG_W_Clear;
            this.state.sig_2ws_src = SIG_W_Stop;
            this.state.sig_2e_src = SIG_E_Stop;
            this.state.sig_2es_src = SIG_E_Stop;
        }
        // East Bound Signals
        else {
            this.state.sig_2w_src = SIG_W_Stop;
            this.state.sig_2ws_src = SIG_W_Stop;
            this.state.sig_2e_src = SIG_E_Stop;
            this.state.sig_2es_src = SIG_E_Clear;
        }
    }
}
else if (this.state.routes[i] === "W_3_1__|
__1_hall_hudson" || this.state.routes[i] === "E_1_3__|
__1_hudson_nysw") {
    // Set Tail Track Colors
    this.state.tail_1_w = color;
    this.state.tail_nysw = color;

    // The Route Is Occupied
    if (this.state.occupied) {
        // Switches
        this.state.sw_1_src = SW_U_W_Occupied;
        this.state.sw_3_src = SW_D_E_R_Occupied;

        // Signals
        this.state.sig_2w_src = SIG_W_Stop;
        this.state.sig_2ws_src = SIG_W_Stop;
        this.state.sig_2e_src = SIG_E_Stop;
        this.state.sig_2es_src = SIG_E_Stop;
    }
    // The Route Is NOT Occupied
    else {
        // Switches

```

```

        this.state.sw_1_src = SW_U_W_Lined;
        this.state.sw_3_src = SW_D_E_R_Lined;

        // Signals
        // West Bound Signals
        if (this.state.routes[i] === "W_3_1__|
__1_hall_hudson") {
            this.state.sig_2w_src = SIG_W_Stop;
            this.state.sig_2ws_src = SIG_W_Clear;
            this.state.sig_2e_src = SIG_E_Stop;
            this.state.sig_2es_src = SIG_E_Stop;
        }
        // East Bound Signals
        else {
            this.state.sig_2w_src = SIG_W_Stop;
            this.state.sig_2ws_src = SIG_W_Stop;
            this.state.sig_2e_src = SIG_E_Clear;
            this.state.sig_2es_src = SIG_E_Stop;
        }
    }
    else if (this.state.routes[i] === "W_3_2__|
__2_hall_hudson" || this.state.routes[i] === "E_2_3__|
__1_hudson_nysw") {
        // Set Tail Track Colors
        this.state.tail_2_w = color;
        this.state.tail_nysw = color;

        // The Route Is Occupied
        if (this.state.occupied) {
            // Switches
            this.state.sw_1_src = SW_U_W_R_Occupied;
            this.state.sw_3_src = SW_D_E_R_Occupied;

            // Signals
            this.state.sig_2w_src = SIG_W_Stop;
            this.state.sig_2ws_src = SIG_W_Stop;
            this.state.sig_2e_src = SIG_E_Stop;
            this.state.sig_2es_src = SIG_E_Stop;
        }
        // The Route Is NOT Occupied
        else {
            // Switches
            this.state.sw_1_src = SW_U_W_R_Lined;
            this.state.sw_3_src = SW_D_E_R_Lined;

            // Signals
            // West Bound Signals
            if (this.state.routes[i] === "W_3_2__|
__2_hall_hudson") {

```

```

        this.state.sig_2w_src = SIG_W_Stop;
        this.state.sig_2ws_src = SIG_W_Clear;
        this.state.sig_2e_src = SIG_E_Stop;
        this.state.sig_2es_src = SIG_E_Stop;
    }
    // East Bound Signals
    else {
        this.state.sig_2w_src = SIG_W_Stop;
        this.state.sig_2ws_src = SIG_W_Stop;
        this.state.sig_2e_src = SIG_E_Stop;
        this.state.sig_2es_src = SIG_E_Clear;
    }
}
}
}
}
}
// ---- END set_route_drawings() ----

/**
 * set_switch_img()
 * @summary Changes image sources for the switches, depending on
switch status
 *
 * This function uses the data passed in through status from the
CTC classes and
 * shows if the switches are reversed or not on the screen, by
changing the image
 * source files, to the correct .png file respectively
 */
set_switch_img() {
    // Set SW #1
    // SW #1 Reversed
    if (this.state.sw_1) {
        this.state.sw_1_src = SW_U_W_R;
    }
    // SW #1 Normal
    else {
        this.state.sw_1_src = SW_U_W;
    }

    // Set SW #3
    // SW #3 Reversed
    if (this.state.sw_3) {
        this.state.sw_3_src = SW_D_E_R;
    }
    // SW #3 Normal
    else {
        this.state.sw_3_src = SW_D_E;
    }
}
}

```



```

// ---- END set_switch_img() ----

/**
 * @summary Function to reset the signal images and track colors
 *
 * This function is need, because if the player was to remove a
route,
 * or when the train clears the interlocking nothing will clear
the route
 * the is displaying on the screen, even if it's gone in the
backend
 */
reset_drawings() {
    this.state.tail_1_w = Empty;
    this.state.tail_2_w = Empty;
    this.state.tail_e = Empty;
    this.state.tail_nysw = Empty;

    this.state.sig_2w_src = SIG_W;
    this.state.sig_2ws_src = SIG_W;
    this.state.sig_2e_src = SIG_E;
    this.state.sig_2es_src = SIG_E;
}
//---- END reset_drawings() ----
}

// Export the interlocking to be drawn on the screen
export default HudsonJunction;

```