Proiect IBM Summer School WebDev Team – [I.E.A]

I. Team Details

Team – [NAME OF TEAM HERE]			
Lead	Dev	Dev	Ops
Andra Horhocea	Andra Gavriliu	Bianca Streuneanu	Zaharia Rafael

Lead: Andra Horhocea was chosen for this role thanks to her technical knowledge, leadership and engagement with the project.

Devs : Andra Gavriliu and Bianca Streuneanu show strong problem-solving skills and attention to detail, two skills really needed for a good developer.

Ops: Zaharia Rafael has good communication skills, which allows him to gather information from the devs, and transform it into a descriptive document to further enhance the team's workflow.

II. Changelog

CHANGELOG		
DATE – [DATA CURSULUI]	LIST OF CHANGES	AUTHOR
[10/07/24]	In the first meeting we learned how to use react-redux and @reduxjs/toolkit in order to make a simple website.	[@everyone]
11/07/24	[Homework:devs used mapbase in order to make a grid]	[Bianca Streuneanu, Andra Horhocea, Andra Gavriliu]
12/07/2024	Modified buttons,rearrengement of controller buttons	Bianca Streuneanu,Andra Horhocea
12/07/2024 (curs 2)	Create a blue border at the edges, the red square should not be able to enter blue boxes. Added a tree image to borders, and a self moving cell (NPC)	@everyone

17/07/2024	Homework: added variables for HP and strenght, and displayed them on the screen, made the buttons <divs>,and made the player into a duck, that moves in 4 directions.(has different images for up, down,left,right)	@Andra Horhocea, Bianca Streuneanu
17/07/2024	Implementations: keyboard controls, animation for npc and attack modal. Changes: Updated UI.	@everyone
22/07/2024 HW	Added animated title, similar spriting method to NPC,attack modal.	Andra Horhocea, Bianca Streuneanu
25/07/2024	Added animation for attack and defend modal	@Andra H
27/07/2024	Applying duck pattern for redux	@Andra H

III. Deep Dive

10/07/2024-11/07/2024

For the first course, we followed a tutorial that explained the basics of Redux, a state management library for React. As discussed in the table above, we initially introduced map base, which was implemented by the devs.

This code renders a table-based map with a highlighted cell in the middle.

MapBase.jsx

src > components > MapBase.jsx > MapBase > renderTable

```jsx
// src/components/MapBase.js
import React from "react";
import { connect } from "react-redux";
import "./../styles/MapBase.css"; // Adjust the path as needed

const MapBase = ({ x, y }) => {
  const renderTable = () => {
    const table = [];
    for (let row = 0; row < 10; row++) {
      const cells = [];
      for (let col = 0; col < 10; col++) {
        const className = (col === 4 && row === 4) ? "red-cell" : "map-cell";
        cells.push(
          <td key={col} className={className}>
            {x === col && y === row ? "P" : ""}
          </td>
        );
      }
      table.push(<tr key={row}>{cells}</tr>);
    }
    return table;
  };

  return (
    <div>
      <p className="map-title">Map</p>
      <table className="map-table">
        <tbody>{renderTable()}</tbody>
      </table>
    </div>
  );
};

const mapStateToProps = (state) => ({
  x: state.x,
  y: state.y,
});
```

src > components > Counter.jsx > ...

```jsx
const Counter = ({ x ,y , incrementX, decrementX,incrementY, decrementY}) => {
  console.log(x,y);
  return (
    <div>
      <p className="counter_title">Player position: X={x} Y={y}</p>
      <button className="button" onClick={decrementY}>
        UP
      </button>
      <button className="button" onClick={decrementX}>
        LEFT
      </button>
      <button className="button" onClick={incrementY}>
        DOWN
      </button>
      <button className="button" onClick={incrementX}>
        RIGHT
      </button>
    </div>
  );
};

const mapStateToProps = (state) => ({
  //  Use 'counter: state.counter.counter' and replace the above line if you are using combineReducers to ensure that 'counter' matches the correct
  x: state.x,
  y: state.y

});

const mapDispatchToProps = (dispatch) => ({
  incrementX: () => dispatch({ type: "RIGHT" }),
  decrementX: () => dispatch({ type: "LEFT" }),
  incrementY: () => dispatch({ type: "UP" }),
  decrementY: () => dispatch({ type: "DOWN" })
});

export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript JSX

This counter displays the player's coordinates and provides buttons to increment or decrement these coordinates.

Counter.jsx index.js

src > reducers > index.js > ...

```
1   const initialState = {
2     x: 0,
3     y: 0
4   };
5
6   const counterReducer = (state = initialState, action) => {
7     switch (action.type) {
8       case "DOWN":
9         return { ...state, y: Math.max(state.y - 1, 0) };
10      case "UP":
11        return { ...state, y: Math.min(state.y + 1, 9) };
12      case "LEFT":
13        return { ...state, x: Math.max(state.x - 1, 0) };
14      case "RIGHT":
15        return { ...state, x: Math.min(state.x + 1, 9) };
16      default:
17        return state;
18    }
19  };
20
21  export default counterReducer;
22
```

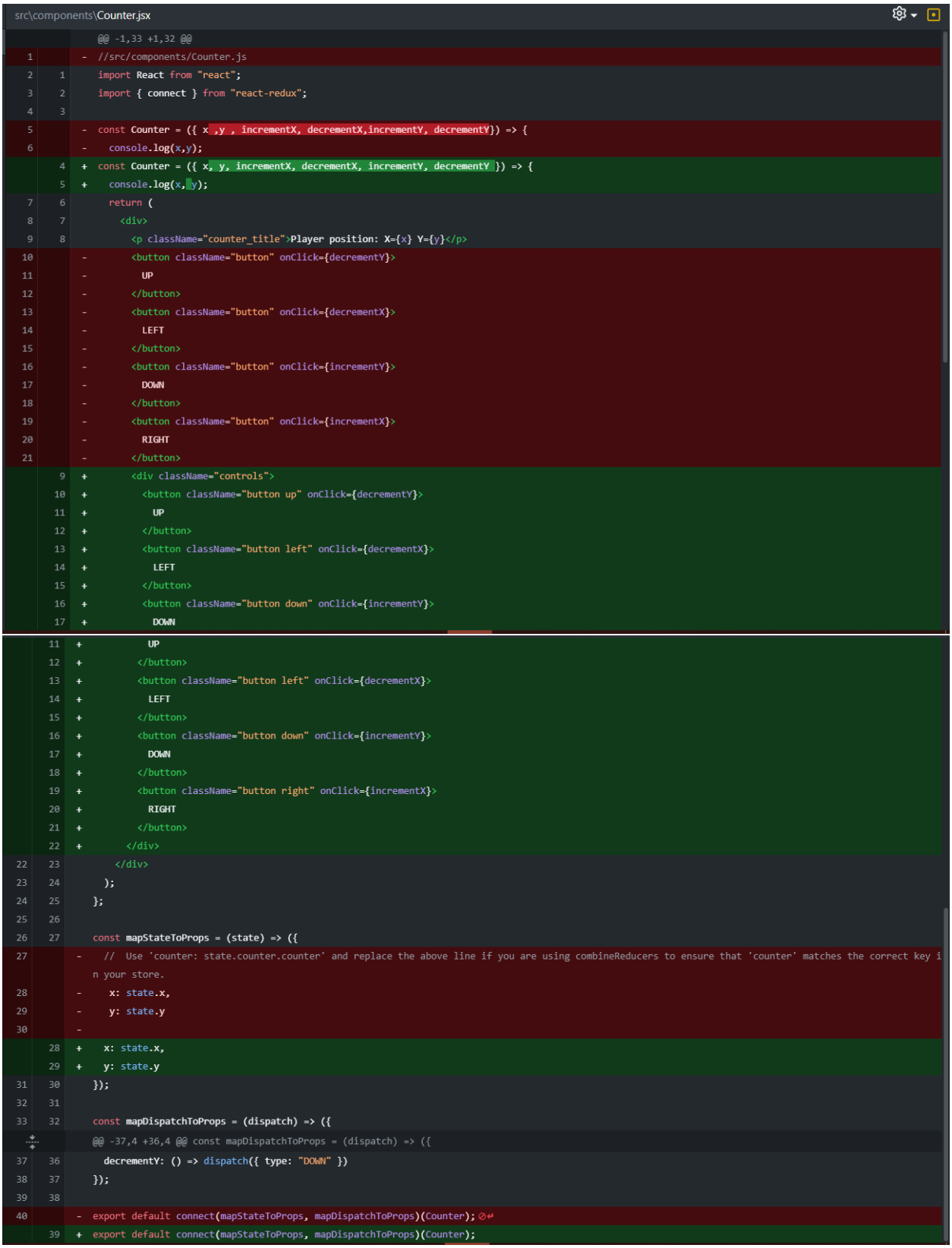
This reducer listens for specific actions : up, down, left, right; and ensures that the coordinates stay within the 0-9 range.

12/07/2024

Bianca streuneanu modified buttons :

```
+
+  .controls {
+    display: grid;
+    grid-template-areas:
+      ". up ."
+      "left . right"
+      ". down .";
+    gap: 5px;
+    justify-items: center;
+    align-items: center;
+  }
+
+  .button.up {
+    grid-area: up;
+  }
+
+  .button.left {
+    grid-area: left;
+  }
+
+  .button.down {
+    grid-area: down;
+  }
+
+  .button.right {
+    grid-area: right;
+  }
```

src\components\Counter.jsx



```
@@ -1,33 +1,32 @@
- //src/components/Counter.js
  import React from "react";
  import { connect } from "react-redux";

- const Counter = ({ x ,y , incrementX, decrementX,incrementY, decrementY}) => {
-   console.log(x,y);
+ const Counter = ({ x, y, incrementX, decrementX, incrementY, decrementY }) => {
+   console.log(x,y);
    return (
      <div>
        <p className="counter_title">Player position: X={x} Y={y}</p>
-       <button className="button" onClick={decrementY}>
-         UP
-       </button>
-       <button className="button" onClick={decrementX}>
-         LEFT
-       </button>
-       <button className="button" onClick={incrementY}>
-         DOWN
-       </button>
-       <button className="button" onClick={incrementX}>
-         RIGHT
-       </button>
+       <div className="controls">
+         <button className="button up" onClick={decrementY}>
+           UP
+         </button>
+         <button className="button left" onClick={decrementX}>
+           LEFT
+         </button>
+         <button className="button down" onClick={incrementY}>
+           DOWN
+         </button>
+         <button className="button right" onClick={incrementX}>
+           RIGHT
+         </button>
+       </div>
      </div>
    );
  };

  const mapStateToProps = (state) => ({
-   //  Use 'counter: state.counter.counter' and replace the above line if you are using combineReducers to ensure that 'counter' matches the correct key in your store.
-   x: state.x,
-   y: state.y
+   x: state.x,
+   y: state.y
  });

  const mapDispatchToProps = (dispatch) => ({
    decrementY: () => dispatch({ type: "DOWN" })
  });

- export default connect(mapStateToProps, mapDispatchToProps)(Counter);
+ export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

-the buttons are now closer to each other, in a cross shape

Andra Horhocea also helped rearranging buttons:

src\App.jsx

```
@@ -1,6 +1,6 @@
// src/App.js
import React from "react";
- import Counter from "./components/Counter";
+ import PlayerController from "./components/PlayerController";
import "./styles/App.css";
import MapBase from "./components/MapBase";

@@ -8,10 +8,10 @@
  return (
    <div className="App">
      <MapBase />
-     <Counter />
+     <PlayerController />
    </div>
  );
}

export default App;
```

src\components\Counter.jsx → src\components\PlayerController.jsx

```
@@ -1,7 +1,8 @@
import React from "react";
import { connect } from "react-redux";
+ import "./../styles/PlayerController.css";

- const Counter = ({ x, y, incrementX, decrementX, incrementY, decrementY }) => {
+ const PlayerController = ({ x, y, incrementX, decrementX, incrementY, decrementY }) => {
  console.log(x, y);
  return (
    <div>

@@ -36,4 +37,4 @@ const mapDispatchToProps = (dispatch) => ({
  decrementY: () => dispatch({ type: "DOWN" })
});

- export default connect(mapStateToProps, mapDispatchToProps)(Counter);
+ export default connect(mapStateToProps, mapDispatchToProps)(PlayerController);
```

src\reducers\index.js

```
@@ -3,7 +3,7 @@ const initialState = {
  y: 0
};

- const counterReducer = (state = initialState, action) => {
+ const playerControllerReducer = (state = initialState, action) => {
  switch (action.type) {
    case "DOWN":
      return { ...state, y: Math.max(state.y - 1, 0) };
@@ -18,4 +18,4 @@ const counterReducer = (state = initialState, action) => {
  }
};

- export default counterReducer;
+ export default playerControllerReducer;
```

src\store\store.js

```
@@ -1,9 +1,9 @@
// src/store/store.js
import { configureStore } from '@reduxjs/toolkit';
- import counterReducer from '../reducers/index';
+ import playerControllerReducer from '../reducers/index';

const store = configureStore({
-   reducer: counterReducer,
+   reducer: playerControllerReducer,
});
```

src\styles\App.css

```
@@ -44,29 +44,3 @@
  color: #888;
}

- .controls {
-   display: grid;
-   grid-template-areas:
-     ". up ."
-     "left . right"
-     ". down .";
-   gap: 5px;
-   justify-items: center;
-   align-items: center;
- }
-
- .button.up {
-   grid-area: up;
- }
-
- .button.left {
-   grid-area: left;
- }
-
- .button.down {
-   grid-area: down;
- }
-
- .button.right {
-   grid-area: right;
- }⊘↵
```

src\styles\PlayerController.css

```
@@ -0,0 +1,29 @@
+ .button.up {
+     grid-area: up;
+   }
+
+   .button.left {
+     grid-area: left;
+   }
+
+   .button.down {
+     grid-area: down;
+   }
+
+   .button.right {
+     grid-area: right;
+   }
+
+   .controls {
+     width: 200px;
+     margin-left: auto;
+     margin-right: auto;
+     gap: 5px;
+     display: grid;
+     grid-template-areas:
+     ". up ."
+     "left . right"
+     ". down .";
+   }
+
+
```

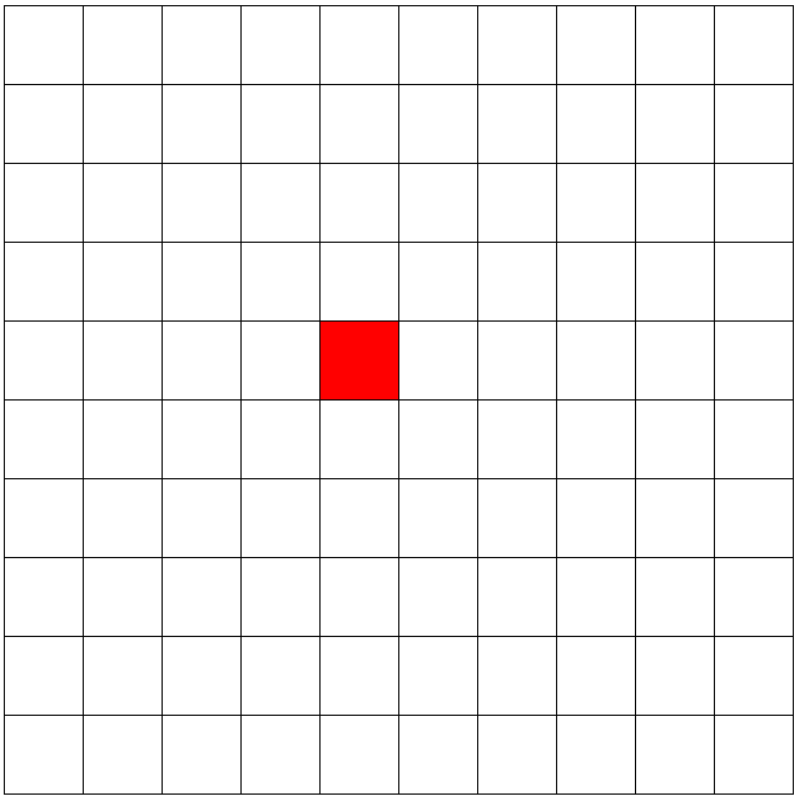
Andra helped changing the player:

src\components\MapBase.jsx

```
@@ -1,4 +1,4 @@
- // src/components/MapBase.js
+ // src/components/MapBase.js
  import React from "react";
  import { connect } from "react-redux";
  import "./../styles/MapBase.css"; // Adjust the path as needed
@@ -9,10 +9,9 @@ const MapBase = ({ x, y }) => {
      for (let row = 0; row < 10; row++) {
        const cells = [];
        for (let col = 0; col < 10; col++) {
-         const className = (col === 4 && row === 4) ? "red-cell" : "map-cell";
+         const className = (col === x && row === y) ? "red-cell" : "map-cell";
          cells.push(
            <td key={col} className={className}>
-             {x === col && y === row ? "P" : ""}
            </td>
          );
        }
```

Final result:

Map



Player position: X=4 Y=4

UP

LEFT RIGHT

DOWN

12/07/2024 Second course

Task 1: Make a blue border that doesn't allow the red square to pass.

-added blue border cells

```
+        for (let col = 0; col < 12; col++) {
+          let className = "map-cell";
+
+          // Adding the blue border cells
+          if (row === 0 || row === 11 || col === 0 || col === 11) {
+            className = "blue-border";
+          }
+
+          // Adding the red player cell
+          if (col === x && row === y) {
+            className = "red-cell";
+          }
+
```

-made sure the player cannot move past the border

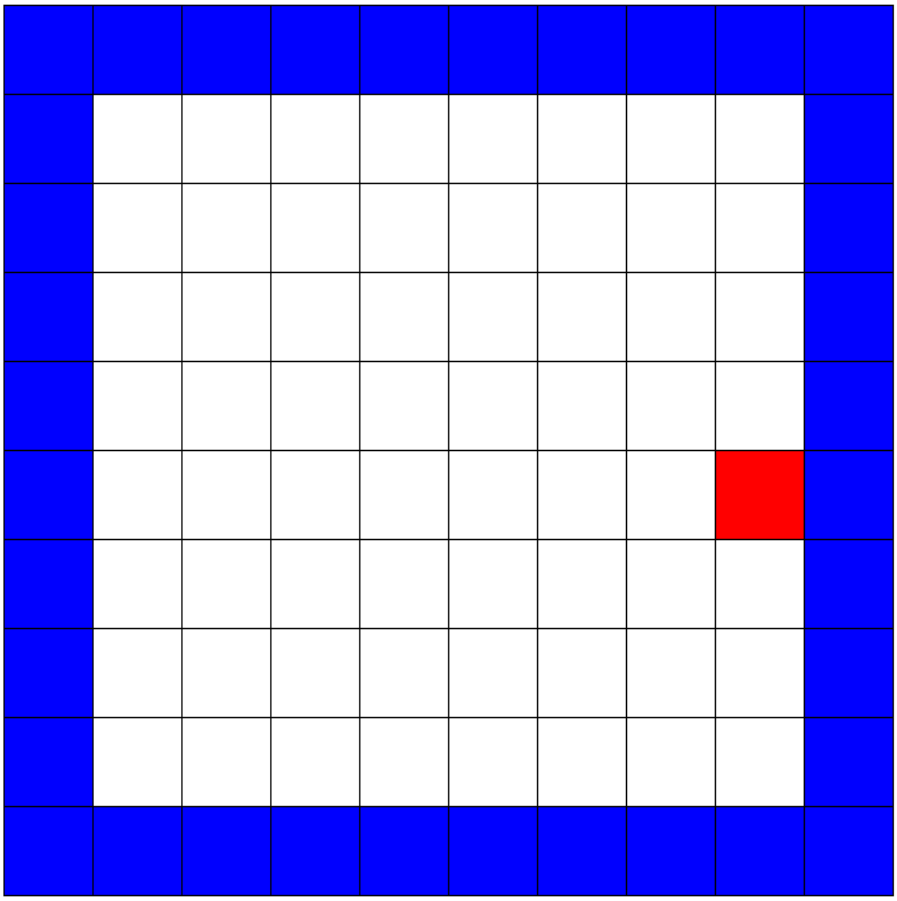
```
+  const mapStateToProps = (state) => {
+    // Ensuring the player can't move past the blue border
+    let x = state.x;
+    let y = state.y;
+    if (x < 1) x = 1;
+    if (x > 10) x = 10;
+    if (y < 1) y = 1;
+    if (y > 10) y = 10;
+
+    return { x, y };
+  };
```

-changed margin colors

```
-            <td key={col} className={className}></td>
+            <td key={col === 0} className={className}
+            style={isBorderCell ? { backgroundColor: "blue" } : {}}
+            ></td>
```

Progress:

Map



Player position: X=8 Y=5

UP

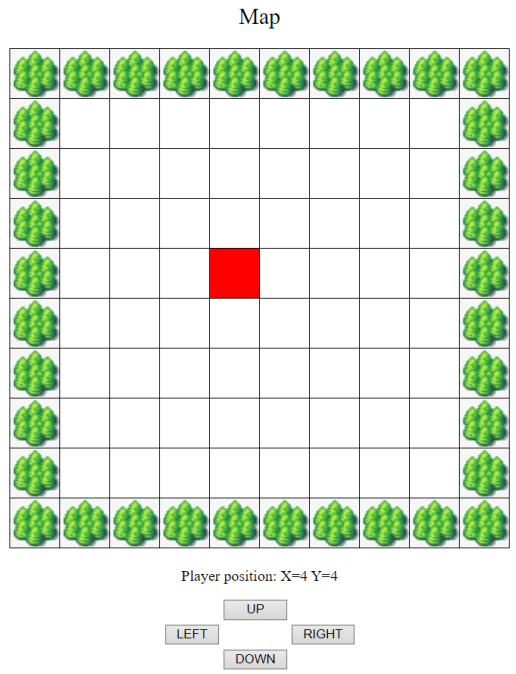
LEFT RIGHT

DOWN

Task 2: Apply image to margins.

```
  import React from "react";
  import { connect } from "react-redux";
  import "./../styles/MapBase.css"; // Adjust the path as needed
+ import marginImage from "./../assets/marginImage.jpeg"; // Adjust the path as needed

  const MapBase = ({ x, y }) => {

  @@ -13,8 +14,8 @@ const MapBase = ({ x, y }) => {
        const isBorderCell = row === 0 || row === 9 || col === 0 || col === 9;
        const className = (col === x && row === y) ? "red-cell" : "map-cell";
        cells.push(
-         <td key={col === 0} className={className}
-         style={isBorderCell ? { backgroundColor: "blue" } : {}}
+         <td key={`${row}-${col}`} className={className}
+         style={isBorderCell ? { backgroundImage: `url(${marginImage})`, backgroundSize: 'cover' } : {}}
          ></td>
        );
      }
```

Progress:



Task 3 : Add NPC (self moving cell)

-changes in MapBase.jsx (state for npc's position, function for it to move randomly)

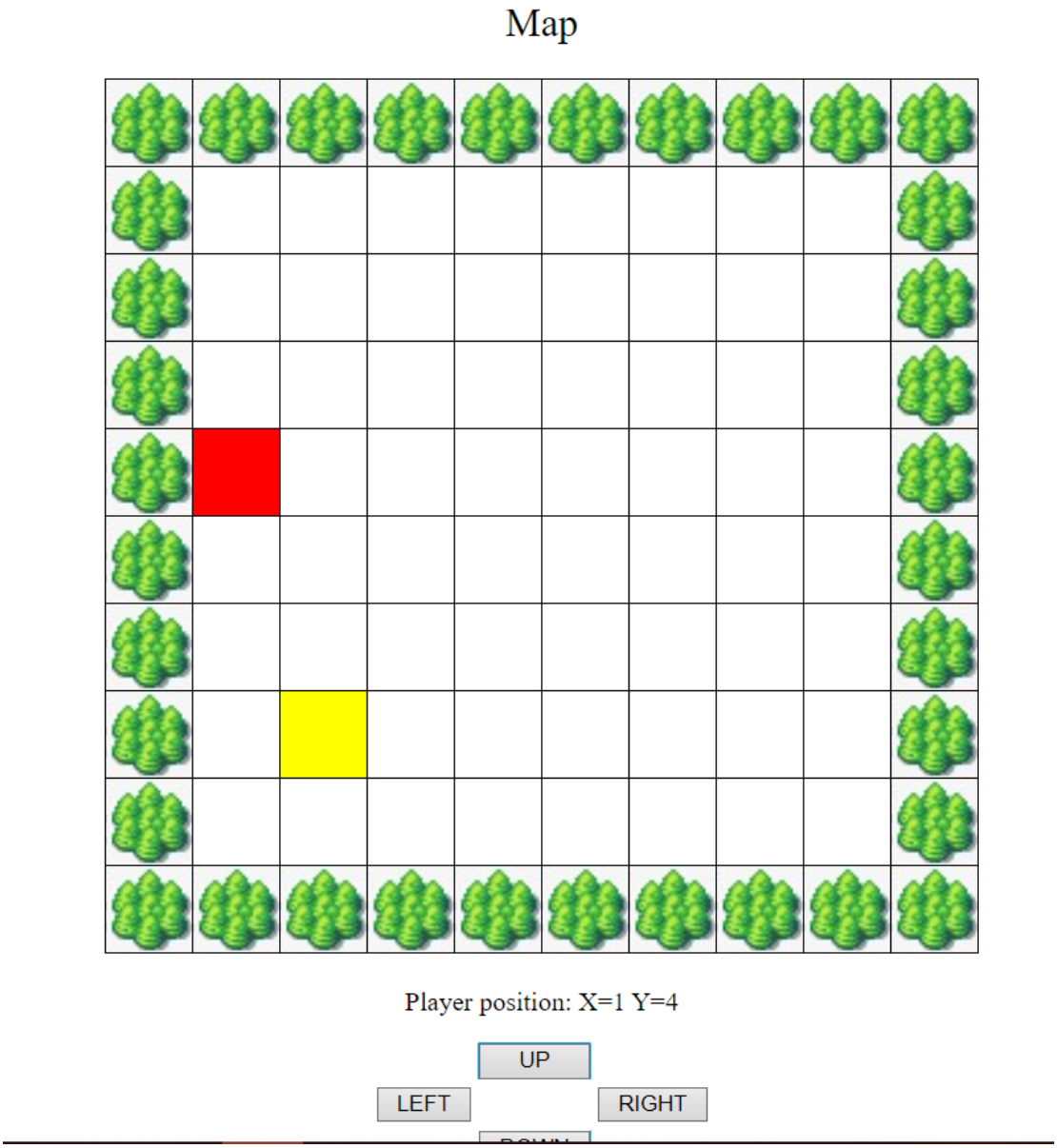
```
+  // State for the yellow dot's position
+  const [yellowDot, setYellowDot] = useState({ x: 1, y: 1 });
+
+  // Function to move the yellow dot randomly
+  const moveYellowDot = () => {
+    const possibleMoves = [
+      { x: 0, y: -1 }, // Up
+      { x: 0, y: 1 },  // Down
+      { x: -1, y: 0 }, // Left
+      { x: 1, y: 0 },  // Right
+    ];
+
+    const currentMove = possibleMoves[Math.floor(Math.random() * possibleMoves.length)];
+
+    setYellowDot((prev) => {
+      const newX = prev.x + currentMove.x;
+      const newY = prev.y + currentMove.y;
+
+      // Ensure the yellow dot stays within the boundaries (1 to 8)
+      if (newX >= 1 && newX <= 8 && newY >= 1 && newY <= 8) {
+        return { x: newX, y: newY };
+      }
+      return prev;
+    });
+  };
+
+  // Update yellow dot's position every second
+  useEffect(() => {
+    const interval = setInterval(moveYellowDot, 1000);
+    return () => clearInterval(interval);
+  }, []);
```

```
15    -          const className = (col === x && row === y) ? "red-cell" : "map-cell";
      45 +       let className = "map-cell";
      46 +
      47 +       if (col === x && row === y) {
      48 +         className = "red-cell";
      49 +       } else if (col === yellowDot.x && row === yellowDot.y) {
      50 +         className = "yellow-cell";
      51 +       }
      52 +
16    53         cells.push(
17    -            <td key={`${row}-${col}`} className={className}
18    -            style={isBorderCell ? { backgroundImage: `url(${marginImage})`, backgroundSize: 'cover' } : {}}
      54 +          <td
      55 +            key={`${row}-${col}`}
      56 +            className={className}
      57 +            style={isBorderCell ? { backgroundImage: `url(${marginImage})`, backgroundSize: 'cover' } : {}}
```

-changes in MapBase.css

```
+ .map-cell {
+   width: 50px;
+   /* Adjust the width as needed */
+   height: 50px;
+   /* Adjust the height as needed */
+   border: 1px solid black;
+   text-align: center;
+   vertical-align: middle;
+ }
+
+ .red-cell {
+   background-color: red;
+ }
+
+ .map-title {
+   text-align: center;
+   font-size: 24px;
+   margin-bottom: 10px;
+ }
+
+ .yellow-cell {
+   background-color: yellow;
+ }⊘↵
```

Progress:



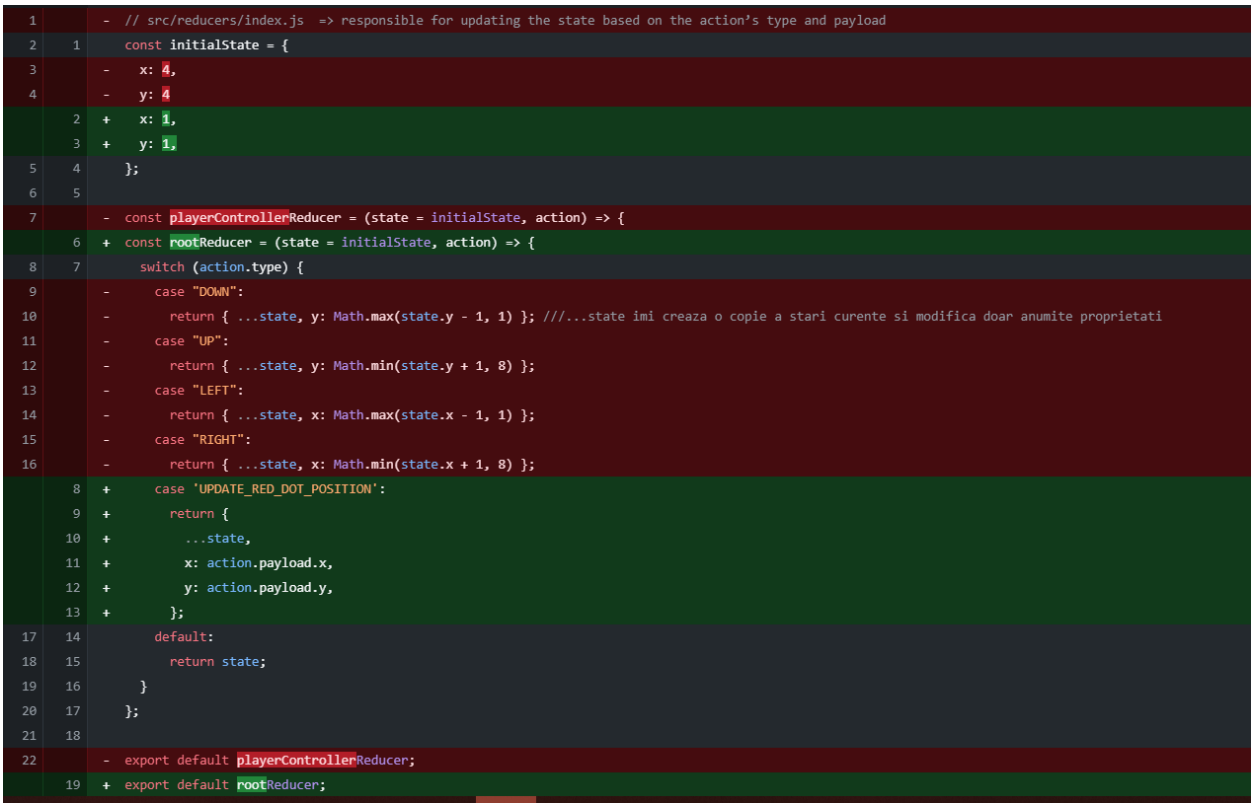
-Bianca Streuneanu added keyboard controls :

-changes in mapBase.jsx:

```
+   // Function to handle key presses and move the red dot
+   const handleKeyPress = (event) => {
+     const { key } = event;
+     let newX = x;
+     let newY = y;
+
+     if (key === "ArrowUp") {
+       newY = y > 1 ? y - 1 : y;
+     } else if (key === "ArrowDown") {
+       newY = y < 8 ? y + 1 : y;
+     } else if (key === "ArrowLeft") {
+       newX = x > 1 ? x - 1 : x;
+     } else if (key === "ArrowRight") {
+       newX = x < 8 ? x + 1 : x;
+     }
+
+     updateRedDotPosition(newX, newY);
+   };
+
+   // Add event listener for key presses
+   useEffect(() => {
+     window.addEventListener("keydown", handleKeyPress);
+     return () => {
+       window.removeEventListener("keydown", handleKeyPress);
+     };
+   }, [x, y]);
+
```

```
- export default connect(mapStateToProps)(MapBase);
+ const mapDispatchToProps = (dispatch) => ({
+   updateRedDotPosition: (x, y) => dispatch({ type: 'UPDATE_RED_DOT_POSITION', payload: { x, y } }),
+ });
+
+ export default connect(mapStateToProps, mapDispatchToProps)(MapBase);
```

-changes to index.js:

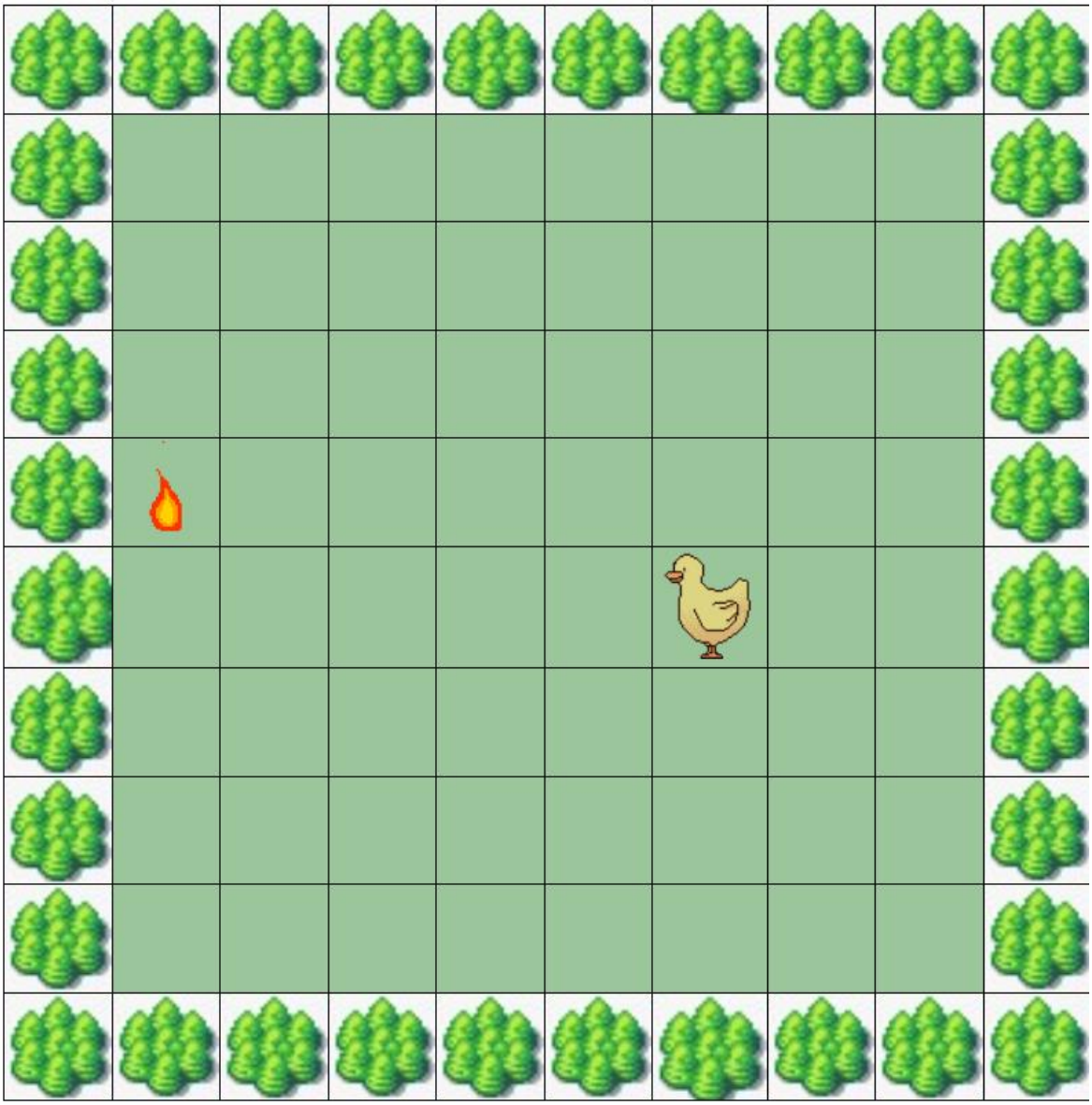


```
- // src/reducers/index.js  => responsible for updating the state based on the action's type and payload
  const initialState = {
-   x: 4,
-   y: 4
+   x: 1,
+   y: 1,
  };

- const playerControllerReducer = (state = initialState, action) => {
+ const rootReducer = (state = initialState, action) => {
    switch (action.type) {
-     case "DOWN":
-       return { ...state, y: Math.max(state.y - 1, 1) }; ///...state imi creaza o copie a stari curente si modifica doar anumite proprietati
-     case "UP":
-       return { ...state, y: Math.min(state.y + 1, 8) };
-     case "LEFT":
-       return { ...state, x: Math.max(state.x - 1, 1) };
-     case "RIGHT":
-       return { ...state, x: Math.min(state.x + 1, 8) };
+     case 'UPDATE_RED_DOT_POSITION':
+       return {
+         ...state,
+         x: action.payload.x,
+         y: action.payload.y,
+       };
      default:
        return state;
    }
  };

- export default playerControllerReducer;
+ export default rootReducer;
```

Homework 17/07/2024

Task : add variables for HP and Strength, and display them on the screen, make the buttons <divs>, modify the player so it no longer is a red square.

First things first, we made the red square a duck, and the yellow square a flame. For the duck we implemented 4 directional sprites, (up,down,left,right) :

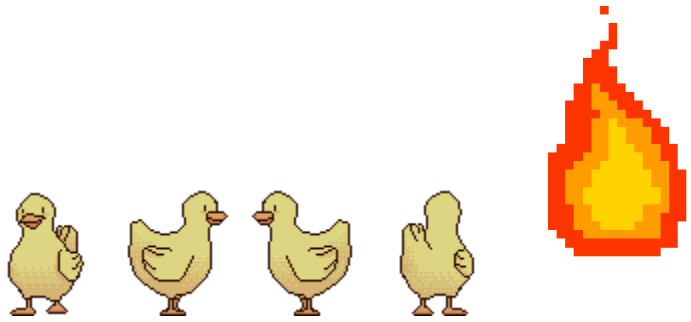


HP: ♥♥♥

Strength:
10

```
4     -  import marginImage from "./../assets/marginImage.jpeg"; // Ad
      4  + import marginImage from "./../assets/marginImage.jpeg";
      5  + import duckImageUp from "./../assets/duck_up.gif";
      6  + import duckImageDown from "./../assets/duck_down.gif";
      7  + import duckImageLeft from "./../assets/duck_left.gif";
      8  + import duckImageRight from "./../assets/duck_right.gif";
5     9
6     -  const MapBase = ({ x, y }) => {
     10  + const MapBase = ({ x, y, direction }) => {
```

Then, Andra H made it so the duck turns in the movement direction(if the player moves to the right, the duck image turns right) :



```
+  const getDuckImage = () => {
+    switch (direction) {
+      case "UP":
+        return duckImageUp;
+      case "DOWN":
+        return duckImageDown;
+      case "LEFT":
+        return duckImageLeft;
+      case "RIGHT":
+        return duckImageRight;
+      default:
+        return duckImageDown;
+    }
+  };
+
```

After that, Andra H added variables for HP and strenght, in the form of hearts.

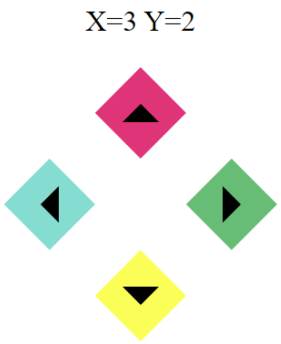
```
+ const PlayerController = ({ x, y, incrementX, decrementX, incrementY, decrementY, hp, strength }) => {
+   const renderHearts = () => {
+     let hearts = [];
+     for (let i = 0; i < hp; i++) {
+       hearts.push(<span key={i} className="heart">♥</span>);
+     }
+     return hearts;
+   };
```

```
+    return (
+      <div className="player-slice">
+        <div className="player-controls">
+          <p className="counter-title"> X={x} Y={y}</p>
+          <div className="controls">
+            <button className="button up" onClick={incrementY}></button>
+            <button className="button left" onClick={decrementX}></button>
+            <button className="button down" onClick={decrementY}></button>
+            <button className="button right" onClick={incrementX}></button>
+          </div>
+        </div>
+        <div className="player-status">
+          <p>HP: {renderHearts()} </p>
+          <p>Strength: {strength} </p>
```

Some little tweaks were made so the duck image and the background fit each other :

```
14  12      vertical-align: middle;
    13  +   background-color: rgb(155, 198, 155);
    14  +
15  15    }
16  16
17      -   .red-cell {
18      -     background-color: red;
    17  +
    18  +
    19  +   .duck-cell {
    20  +     /* background-image: url('./../assets/duck_down.gif'); */
    21  +     background-size: contain; /* Ajustează dimensiunea pentru a se potrivi conținutului */
    22  +     background-color: rgb(155, 198, 155);
19  23    }
20  24
    25  +
21  26    .map-title {
22  27      text-align: center;
23      -     font-size: 24px;
24  28      margin-bottom: 10px;
25  29    }
26  30
27      -   .yellow-cell {
28      -     background-color: yellow;
    31  +   .fire-cell {
    32  +     background-image: url('./../assets/fire.gif');
    33  +     background-size: contain; /* Ajustează dimensiunea pentru a se potrivi conținutului */
    34  +     background-repeat: no-repeat; /* Asigură că imaginea nu se repetă */
    35  +     background-position: center;
    36  +     background-color: rgb(155, 198, 155);
29  37    }
```

The movement buttons were made prettier :



```
@@ -1,29 +1,89 @@
+ .player-slice {
+   display: inline-grid;
+   grid-template-columns: 8;
+   gap: 20%
+ }
+
+ .player-controls {
+   grid-column-start: 2;
+   grid-column-end: 3;
+ }
+
+ .button {
+   width: 50px;
+   height: 50px;
+   border: 2px  black;
+   display: flex; /* pentru centrarea conținutului */
+   justify-content: center; /* centru orizontal al conținutului */
+   align-items: center; /* centru vertical al conținutului */
+   clip-path: polygon(50% 0%, 100% 50%, 50% 100%, 0% 50%); /* formă romb */
+ }
```

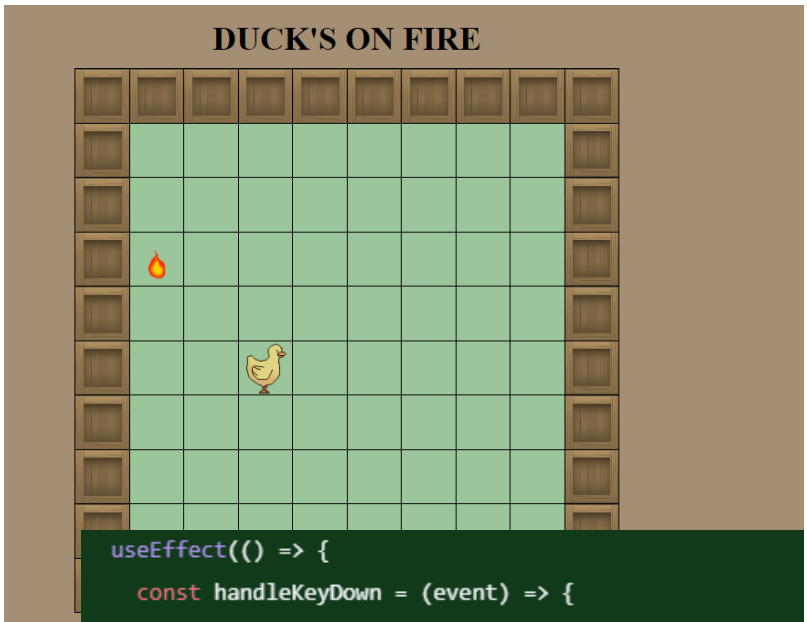
```
  grid-area: up;
  background-color: rgb(255, 0, 119);
}

.button.left {
  grid-area: left;
  background-color: turquoise;
}

.button.down {
  grid-area: down;
  background-color: yellow;
}

.button.right {
  grid-area: right;
  background-color: rgb(19, 192, 108);
```

```
  .button:hover {
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

  }

  .button:active {
    background-color:blueviolet;
  }
```

```
+    .player-status {
+      grid-column-start: 7;
+      grid-column-end: 8;
     }

+  .button::after {
+    content: '';
+    border-style: solid;
+  }
+
+  .button.up::after {
+    border-width: 0 10px 10px 10px;
+    border-color: transparent transparent rgb(0, 0, 0) transparent;
+  }
+
+  .button.left::after {
+    border-width: 10px 10px 10px 0;
+    border-color: transparent black transparent transparent;
+  }
+
+  .button.down::after {
+    border-width: 10px 10px 0 10px;
+    border-color: black transparent transparent transparent;
+  }
+
+  .button.right::after {
+    border-width: 10px 0 10px 10px;
+    border-color: transparent transparent transparent black;
+  }
```

17/07/2024 Third Course

Andra H changed UI:



- ✓ Task 1: Implement keyboard controls:

  -added event listener for keydown:

```
useEffect(() => {
  const handleKeyDown = (event) => {
    switch (event.key) {
      case 'ArrowUp':
        incrementY();
        break;
```

-cleanup event listener:

```
  return () => {
    window.removeEventListener('keydown', handleKeyDown);
  };
}, [incrementX, decrementX, incrementY, decrementY]);
```

22/07/2024 Homework:

-modified opponent cell:

```
        break;
      case 'ArrowRight':
        incrementX();
        break;
      default:
        break;
    }
  };
```

```
-   const [yellowDot, setYellowDot] = useState({ x: 1, y: 1 });
+   const [opponent, setOpponent] = useState({ x: 8, y: 8, direction: "UP" });
```

```
-   const interval = setInterval(moveYellowDot, 1000);
-   return () => clearInterval(interval);
+   const opponentInterval = setInterval(moveOpponent, 1000);
+   return () => {
+     clearInterval(opponentInterval);
+   };
```

```
+     { x: 0, y: -1, direction: "DOWN" },  // Up
+     { x: 0, y: 1, direction: "UP" },     // Down
+     { x: -1, y: 0, direction: "LEFT" },  // Left
+     { x: 1, y: 0, direction: "RIGHT" },  // Right
```

```
+            {className === "opponent-cell" && (
+              <img
+                src={getDuckImage(opponent.direction)}
+                alt="Opponent Duck"
+                className="duck-image duck-opponent-image"
+              />
+            )}
```

```
+    position: absolute;
+    width: 50px;
+    height: 50px;
     background-color: rgb(155, 198, 155);
+    filter:  hue-rotate(-50deg);
```

- Added modal component :
  This displays a dialog modal with ”Attack” and ”Defend” buttons. It accepts handleAttack, show and children as props. The

```
+ import './../styles/modal.css';
+
+ const Modal = ({ handleClose: handleAttack, show, children }) => {
+   const showHideClassName = show ? "modal display-block" : "modal display-none";
+
+   return (
+     <div className={showHideClassName}>
+       <section className="modal-main">
+         {children}
+         <button type="button" onClick={handleAttack}>
+           Attack
+         </button>
+         <button type="button" onClick={handleAttack}>
+           Defend
+         </button>
+       </section>
+     </div>
+   );
+ };
+
+ export default Modal;
```

  showHideClassName variable dynamically sets the CSS class based on the show prop, toggling between "modal display-block" and "modal display-none" to only show the modal when its needed.

  This is the css file.

```
+ .modal {
+     position: fixed;
+     top: 0;
+     left: 0;
+     width:100%;
+     height: 100%;
+     background: rgba(60, 38, 29, 0.6);
+   }
+
+   .modal-main {
+     position:fixed;
+     background-color: rgb(165, 143, 116);
+     width: 80%;
+     height: auto;
+     top:50%;
+     left:50%;
+     transform: translate(-50%,-50%);
+   }
+
+   .display-block {
+     display: block;
+   }
+
+   .display-none {
+     display: none;
+   }
+
```

Added modal component to only show when the player is on the same tile as the enemy:

```
+    useEffect(() => {
+      if (x === opponent.x && y === opponent.y) {
+        setShowModal(true);
+      }
+    }, [x, y, opponent]);
+
```

```
+        <Modal show={showModal} handleClose={() => setShowModal(false)}>
+          <p>The player and the opponent have met!</p>
+        </Modal>
```

Added variables for enemy HP and strenght:

```
+          <p>Player Status</p>
+          <p>HP: {renderHearts(player_hp)} </p>
+          <p>Strength: {player_strength} </p>
        </div>
      </div>
+     </div>
    );
  };

  const mapStateToProps = (state) => ({
    x: state.x,
    y: state.y,
-   hp: state.hp,
-   strength: state.strength,
+   player_hp: state.player_hp,
+   player_strength: state.player_strength,
+   opponent_hp: state.opponent_hp,
+   opponent_strength: state.opponent_strength,
```

```
+ const PlayerController = ({
+   x,
+   y,
+   incrementX,
+   decrementX,
+   incrementY,
+   decrementY,
+   player_hp,
+   player_strength,
+   opponent_hp,
+   opponent_strength,
+   isNpcMovable
+ }) => {
```

```
+    player_hp: 3,
+    player_strength: 10,
+    opponent_hp: 3,
+    opponent_strength: 10,
```

```
<div className="game-slice">
  <div className="opponent-status">
    <p>Opponent Status</p>
    <p>HP: {renderHearts(opponent_hp)}</p>
    <p>Strength: {opponent_strength}</p>
  </div>
```

```
+    grid-column-start: 3;
+    grid-column-end: 7;
+    margin: 0px 15px;
+    background-color: rgba(227, 226, 148, 0.089);
   }


   @@ -85,5 +87,16 @@
     background-color:blueviolet;
   }
+ .opponent-status {
+    display: inline;
+    background-color: aliceblue;
+ }

+ .game-slice {
+    display: flex;
+ }

+ .opponent-status {
+    margin-left: 150px;
+    background-color: rgba(165, 116, 116, 0.192);
+ }
```

Added attack and defend functionality:

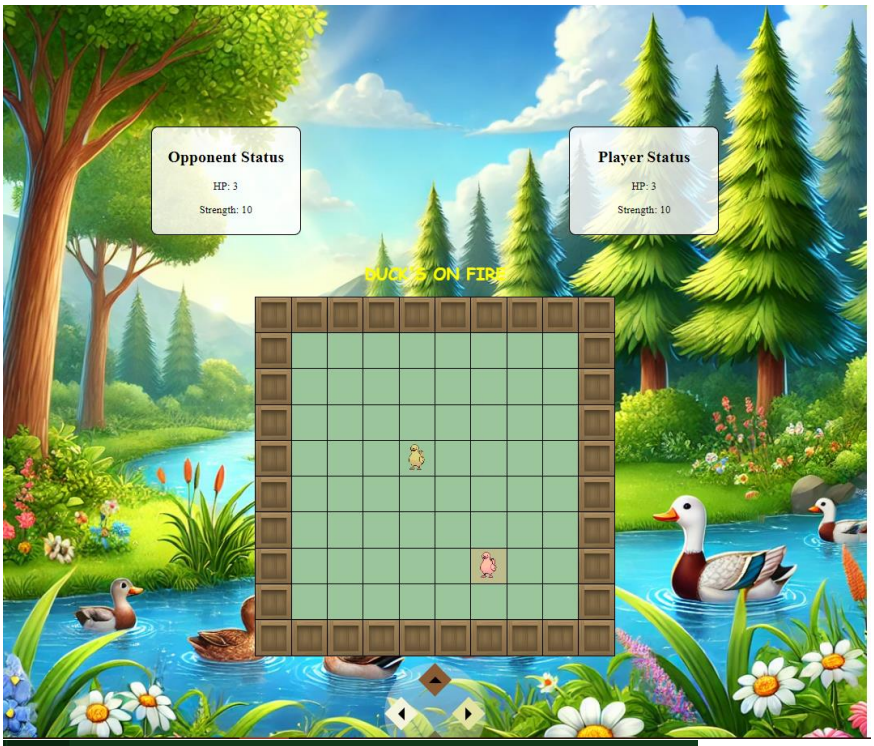
```
+ export const START_ATTACK = "START_ATTACK";
+ export const STOP_ATTACK = "STOP_ATTACK";
+
+ // Action creators
+ export const startAttack = () => ({ type: START_ATTACK });
+ export const stopAttack = () => ({ type: STOP_ATTACK });
```

```
+   const handleAttack = () => {
+     // Poți adăuga orice logică dorești pentru butonul Attack
+     console.log('Attack clicked');
+   };
+
+   const handleDefend = () => {
+     closeModal(); // Închide modalul când apăsăm pe Defend
+   };
+
```

- This code extends the Modal component by adding functionality for dynamic styling and button handling. It introduces modalClassName, which conditionally appends a "red-background" class if isRed is true, enhancing the modal's visual feedback. The handleAttackClick and handleDefendClick functions are defined to trigger the provided handleAttack and handleDefend functions when the respective buttons are clicked, ensuring the correct actions are executed. Additionally, a useEffect hook is employed to toggle the isRed state every 800 milliseconds while the modal is shown, creating a blinking effect. The interval is cleared when the modal is hidden or the component unmounts, ensuring proper cleanup of resources.

```
+   const modalClassName = isRed ? `${showHideClassName} red-background` : showHideClassName;
+
+   const handleAttackClick = () => {
+     if (handleAttack) {
+       handleAttack(); // Execută funcția de atac
+     }
+   };
+
+   const handleDefendClick = () => {
+     if (handleDefend) {
+       handleDefend(); // Execută funcția de apărare
+     }
+   };
+
+   useEffect(() => {
+     if (show) {
+       const interval = setInterval(() => {
+         setIsRed((prev) => !prev); // Comută starea între roșu și normal
+       }, 800); // Alternați la fiecare 3 secunde
+
+       return () => clearInterval(interval); // Curăță intervalul atunci când componenta se dezleagă
+     }
+   }, [show]);
```

Bianca S made the backround prettier :




```
124 +      <div className="map-container">
125 +        <div className="status-container">
126 +          <div className="status-box">
127 +            <h2>Opponent Status</h2>
128 +            <p>HP: {opponent_hp}</p>
129 +            <p>Strength: {opponent_strength}</p>
130 +          </div>
131 +          <div className="status-box">
132 +            <h2>Player Status</h2>
133 +            <p>HP: {player_hp}</p>
134 +            <p>Strength: {player_strength}</p>
135 +          </div>
136 +        </div>
137           <h1 className="map-title">DUCK'S ON FIRE</h1>
138           <table className="map-table">
139             <tbody>{renderTable()}</tbody>
...     @@ -137,7 +149,11 @@ const mapStateToProps = (state) =
149       x: state.x,
150       y: state.y,
151       direction: state.direction,
-       isNpcMovable: state.isNpcMovable
152 +     isNpcMovable: state.isNpcMovable,
153 +     player_hp: state.player_hp,
154 +     player_strength: state.player_strength,
155 +     opponent_hp: state.opponent_hp,
156 +     opponent_strength: state.opponent_strength
```

Bianca Streunanu also modified status hp and strength bar:

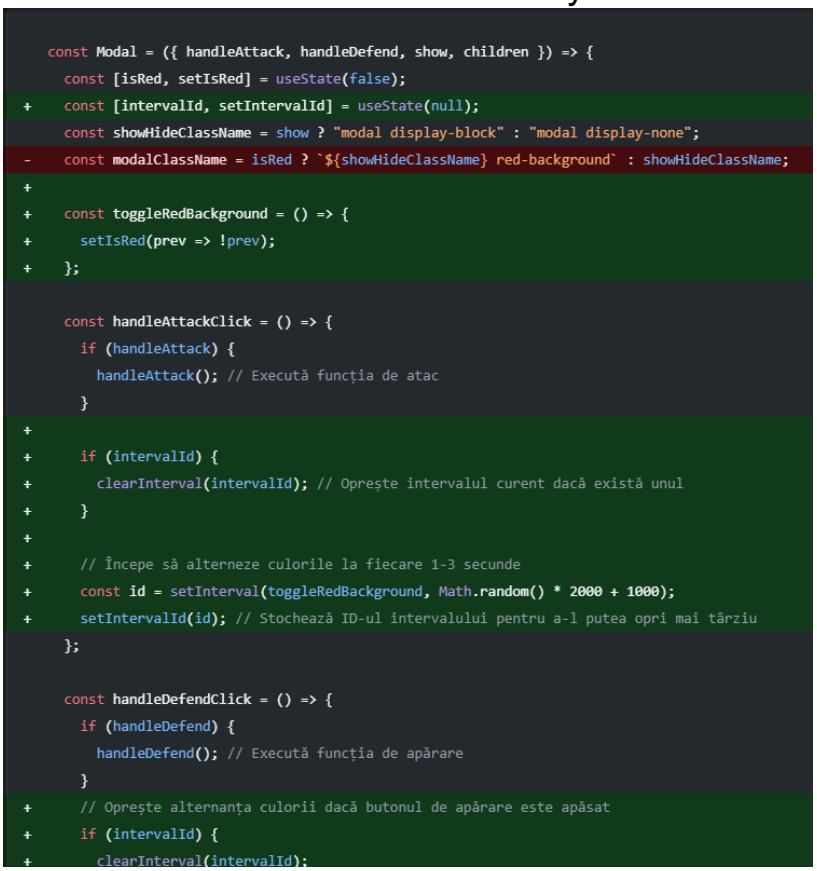
```
+ /* src/styles/App.css */
+
  #root {
-   max-width: 1280px;
-   text-align: center;
+   height: 100%;
+   display: flex;
+   justify-content: center;
+   align-items: center;
  }


-
- * {
-   background-color: rgb(165, 143, 116);
+ .App {
+   width: 100%;
+   max-width: 1280px;
+   text-align: center;
```

```
+   height: 100%;
+   margin: 0;
+   padding: 0;
+ }
+
+ #root {
+   height: 100%;
+   display: flex;
+   justify-content: center;
+   align-items: center;
+ }
+
+ .map-container {
+   background-image: url('../assets/background.png');
+   background-size: cover;
+   background-position: center;
+   width: 100%;
+   height: 100%;
+   display: flex;
+   flex-direction: column;
+   align-items: center;
+   justify-content: center;
+   padding-top: 200px;
+   box-sizing: border-box;
+ }
+
+ .status-container {
+   display: flex;
+   justify-content: space-around;
+   width: 100%;
+   padding: 20px;
+   box-sizing: border-box;
+   margin-bottom: 10px;
```

```
+ .controls-container {
+   display: flex;
+   justify-content: center;
+   margin-top: 20px;
+ }
+
+ .control-button {
+   background-color: rgba(255, 255, 255, 0.8);
+   border: 1px solid black;
+   border-radius: 5px;
+   padding: 10px 20px;
+   margin: 0 5px;
+   cursor: pointer;
+   transition: background-color 0.3s ease;
+ }
+
+ .control-button:hover {
+   background-color: rgba(255, 255, 255, 1);
+ }
```

The CSS code provided defines several styles for elements in the application. The .duck-cell class sets a green background color and includes a smooth transition effect for transformations, which can enhance user experience with visual animations. This ensures that any changes to the .duck-cell element, such as movement or scaling, will occur smoothly over 0.5 seconds.

Andra H. corrected the functionality of the attack button in the modal:



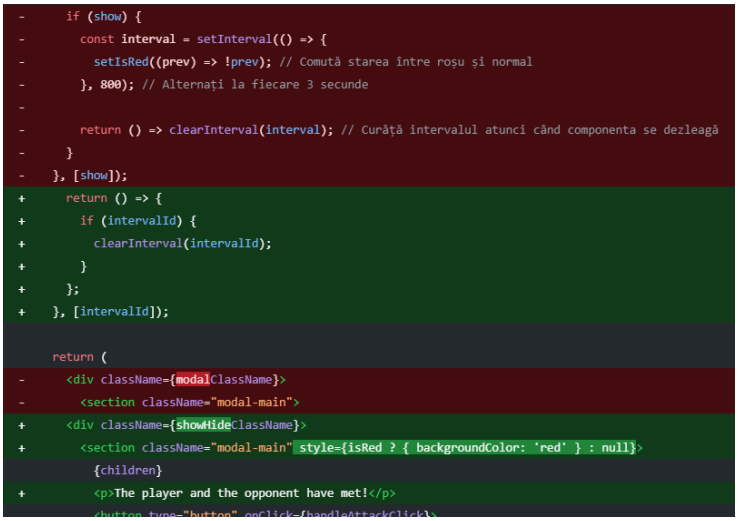
```
const Modal = ({ handleAttack, handleDefend, show, children }) => {
  const [isRed, setIsRed] = useState(false);
+ const [intervalId, setIntervalId] = useState(null);
  const showHideClassName = show ? "modal display-block" : "modal display-none";
- const modalClassName = isRed ? `${showHideClassName} red-background` : showHideClassName;
+
+ const toggleRedBackground = () => {
+   setIsRed(prev => !prev);
+ };

  const handleAttackClick = () => {
    if (handleAttack) {
      handleAttack(); // Execută funcția de atac
    }
+
+   if (intervalId) {
+     clearInterval(intervalId); // Oprește intervalul curent dacă există unul
+   }
+
+   // Începe să alterneze culorile la fiecare 1-3 secunde
+   const id = setInterval(toggleRedBackground, Math.random() * 2000 + 1000);
+   setIntervalId(id); // Stochează ID-ul intervalului pentru a-l putea opri mai târziu
  };

  const handleDefendClick = () => {
    if (handleDefend) {
      handleDefend(); // Execută funcția de apărare
    }
+   // Oprește alternanța culorii dacă butonul de apărare este apăsat
+   if (intervalId) {
+     clearInterval(intervalId);
```

Now, the modal alternates its background colors at random intervals after the attack button is pressed.

Andra H also added a timer :

```
+   const [attackIntervalId, setAttackIntervalId] = useState(null);
+   const [defendIntervalId, setDefendIntervalId] = useState(null);
+   const [isAttackActive, setIsAttackActive] = useState(false);
+   const [isDefendActive, setIsDefendActive] = useState(false);
+   const [attackTimer, setAttackTimer] = useState(0);
+   const [defendTimer, setDefendTimer] = useState(0);
+
    const showHideClassName = show ? "modal display-block" : "modal display-none";

-   const toggleRedBackground = () => {
-     setIsRed(prev => !prev);
+   useEffect(() => {
+     // Cleanup on component unmount or when show changes
+     return () => {
+       clearInterval(attackIntervalId);
+       clearInterval(defendIntervalId);
+     };
+   }, [attackIntervalId, defendIntervalId]);
+
+   const startAttackTimer = () => {
+     if (attackIntervalId) {
+       clearInterval(attackIntervalId);
+     }
+
+     setIsAttackActive(true);
+     setAttackTimer(0);
+
+     const id = setInterval(() => {
+       setIsRed(prev => !prev);
+       setAttackTimer(prev => prev + 1);
+     }, Math.random() * 2000 + 1000); // Interval aleator între 1 și 3 secunde
+
+     setAttackIntervalId(id);
```

```
+       clearInterval(defendIntervalId);
+     }
+
+     setIsDefendActive(true);
+     setDefendTimer(0);
+
+     const id = setInterval(() => {
+       console.log('Defend action running');
+       setDefendTimer(prev => prev + 1);
+     }, 1000);
+
+     setDefendIntervalId(id);
+   };

    return (
      <div className={showHideClassName}>
@@ -55,6 +82,24 @@ const Modal = ({ handleAttack, handleDefend,
          <button type="button" onClick={handleDefendClick}>
            Defend
          </button>
+
+         <div className="timers-info">
+           {isAttackActive && (
+             <div>
+               <p>Attack Timer Active</p>
+               <p>Time Elapsed: {attackTimer} seconds</p>
+             </div>
+           )}
+           {!isAttackActive && <p>Attack Timer Inactive</p>}
+
+           {isDefendActive && (
+             <div>
+               <p>Defend Timer Active</p>
```



25/07/2024

Modal attack and defend colors and other changes

```
+       // setarea culorilor atacului
+       setIsLightGreen(true);
+       setIsGreen(false);
+
+       // setarea duratei culorilor
+       const lightGreenDuration = 1000; // 1 secunda
+       const totalDuration = Math.random() * 2000 + 1000;
+
+       //  interval monitorizare buton de atac pentru setarea culorii inchise
        const id = setInterval(() => {
-         setIsRed(prev => !prev);
+         if (isClicked) {
+           clearInterval(attackIntervalId);
+           setIsLightGreen(false);
+           setIsGreen(true);
+         }
          setAttackTimer(prev => prev + 1);
-       }, Math.random() * 2000 + 1000); // Interval aleator între 1 și 3 secunde
+       }, totalDuration);

        setAttackIntervalId(id);
+
+       setTimeout(() => {
+         if (isClicked) {
+           setIsLightGreen(false);
+           setIsGreen(true);
+         }
+       }, lightGreenDuration);
```

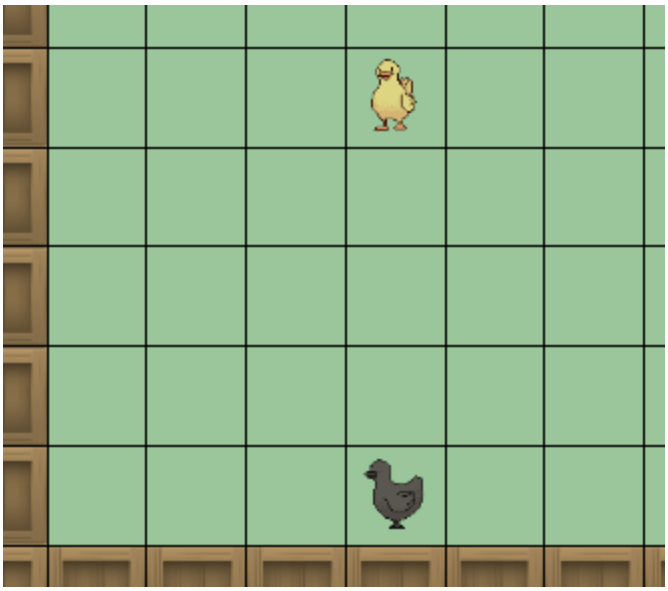
```
+      // Executa functia de atac doar când este verde închis
+      setIsClicked(true);
+      if (isGreen) {
+        if (handleAttack) {
+          handleAttack();
+        }
       }

-      startAttackTimer(); // Începe alternarea culorii roșii
+      // Incepe temporizatorul pentru fazele de culoare
+      startAttackTimer();
     };

+    // Logica pentru timer ul de aparare
     const handleDefendClick = () => {
       if (handleDefend) {
-        handleDefend(); // Execută funcția de apărare
+        // Executa functia de aparare
+        handleDefend();
       }

-      // Oprește temporizatorul de atac
+      // Oprire cronometru atac & resetare culori la initial
       if (attackIntervalId) {
         clearInterval(attackIntervalId);
         setIsAttackActive(false);
-        setIsRed(false); // Revine la culoarea inițială
+        setIsLightGreen(false);
+        setIsGreen(false);
```

27/07/2024

Applied duck pattern for enemy

```
import enemyDuckUpImage from "./../assets/enemy_duck_up.gif";
import enemyDuckDownImage from "./../assets/enemy_duck_down.gif";
import enemyDuckLeftImage from "./../assets/enemy_duck_left.gif";
import enemyDuckRightImage from "./../assets/enemy_duck_right.gif";
import { setNpcMovable } from "../reducers/playerController";
import { moveOpponent } from "../reducers/opponent";
import { showModal, hideModal } from '../reducers/modalSlice.js';
import StatusDisplay from "./StatusDisplay.jsx";

const MapBase = ({
  x, y, direction, isNpcMovable, setNpcMovable, player_hp, player_strength,
  opponent, moveOpponent, showModal, hideModal, showModalState
}) => {
```

```
  }, [x, y, opponent, setNpcMovable, showModal]);

  const getDuckImage = (direction, isPlayer) => {
    if (isPlayer) {
      switch (direction) {
        case "UP":
          return duckUpImage;
        case "DOWN":
          return duckDownImage;
        case "LEFT":
          return duckLeftImage;
        case "RIGHT":
          return duckRightImage;
        default:
          return duckUpImage;
      }
    } else {
      switch (direction) {
        case "UP":
          return enemyDuckUpImage;
        case "DOWN":
          return enemyDuckDownImage;
        case "LEFT":
          return enemyDuckLeftImage;
        case "RIGHT":
          return enemyDuckRightImage;
        default:
          return enemyDuckUpImage;
```



Changed status display:

```jsx
import React from "react";
import { connect } from "react-redux";
import playerImg from "./../assets/duck_up.gif";
import enemyImg from "./../assets/enemy_duck_up.gif";
import { showModal } from '../reducers/modalSlice';


const StatusDisplay = ({ playerHp, playerStrength, opponentHp, opponentStrength }) =
  return (
    <div className="status-container">
      <div className="status-box"><h2>Player Status</h2>
      <p>HP: {playerHp}</p>
      <p>Strength: {playerStrength}</p>
      {showModal &&<img src={playerImg} ></img>}
      </div>
      =<div className="status-box"> <h2>Opponent Status</h2>
      <p>HP: {opponentHp}</p>
      <p>Strength: {opponentStrength}</p>
      {showModal && <img src={enemyImg}></img>}
      </div>
    </div>
  );
};

const mapStateToProps = (state) => ({
  playerHp: state.playerController.player_hp,
  playerStrength: state.playerController.player_strength,
  opponentHp: state.opponent.opponent_hp,
  opponentStrength: state.opponent.opponent_strength
});

export default connect(mapStateToProps)(StatusDisplay);
```

Changed modal slice:

```
+ // src/reducers/modalSlice.js
+
+ // Action Types
+ const SHOW_MODAL = 'modal/SHOW_MODAL';
+ const HIDE_MODAL = 'modal/HIDE_MODAL';
+
+
+ // Initial State
+ const initialState = {
+   showModal: false,
+ };
+
+ // Reducer
+ const modalReducer = (state = initialState, action) => {
+   switch (action.type) {
+     case SHOW_MODAL:
+       return { ...state, showModal: true };
+     case HIDE_MODAL:
+       return { ...state, showModal: false };
+     default:
+       return state;
+   }
+ };
+
+ // Action Creators
+ export const showModal = () => ({ type: SHOW_MODAL });
+ export const hideModal = () => ({ type: HIDE_MODAL });
+
+
+ export default modalReducer;
```

Changes to enemy movement:

```
const MOVE_UP = "playerController/MOVE_UP";
const MOVE_DOWN = "playerController/MOVE_DOWN";
const MOVE_LEFT = "playerController/MOVE_LEFT";
const MOVE_RIGHT = "playerController/MOVE_RIGHT";
const SET_NPC_MOVABLE = "playerController/SET_NPC_MOVABLE";
const SET_PLAYER_HP = "playerController/SET_PLAYER_HP";
const SET_PLAYER_STRENGTH = "playerController/SET_PLAYER_CONTROLLER";


// Stare initiala
const initialState = {
  x: 4,
  y: 4,
  player_hp: 3,
  player_strength: 10,
  direction: "UP",
  isNpcMovable: true,
  isAttacking: false, // Added state to track attack status

};

// Reducer
const playerControllerReducer = (state = initialState, action) => {
  switch (action.type) {
    case MOVE_UP:
      if (!state.isNpcMovable) return state; // Nu permite mișcarea dacă NPC-ul nu este mișcabil
      return { ...state, y: Math.max(state.y - 1, 1), direction: "UP" };
    case MOVE_DOWN:
      if (!state.isNpcMovable) return state; // Nu permite mișcarea dacă NPC-ul nu este mișcabil
      return { ...state, y: Math.min(state.y + 1, 8), direction: "DOWN" };
    case MOVE_LEFT:
      if (!state.isNpcMovable) return state; // Nu permite mișcarea dacă NPC-ul nu este mișcabil
```

```
// Action Creators (creatori de acțiuni)
export const moveUp = () => ({ type: MOVE_UP });
export const moveDown = () => ({ type: MOVE_DOWN });
export const moveLeft = () => ({ type: MOVE_LEFT});
export const moveRight = () => ({ type: MOVE_RIGHT});
export const setNpcMovable = (isMovable) => ({ type: SET_NPC_MOVABLE, payload: isMovable});
export const setPlayerHp = (hp) => ({ type: SET_PLAYER_HP, payload: hp });
export const setPlayerStrength = (strength) => ({ type: SET_PLAYER_STRENGTH, payload: strength });

export default playerControllerReducer;
```

```
+    x: state.playerController.x,
+    y: state.playerController.y,
+    direction: state.playerController.direction,
+    isNpcMovable: state.playerController.isNpcMovable,
+    player_hp: state.playerController.player_hp,
+    player_strength: state.playerController.player_strength,
+    opponent: state.opponent,
+    showModalState: state.modal.showModal
  });

  const mapDispatchToProps = (dispatch) => ({
-    setIsNpcMovable: (movable) => dispatch({ type: "SET_NPC_MOVABLE", payload: movable })
+    setNpcMovable: (movable) => dispatch(setNpcMovable(movable)),
+    moveOpponent: (x, y, direction) => dispatch(moveOpponent(x, y, direction)),
+    showModal: () => dispatch(showModal()),
+    hideModal: () => dispatch(hideModal())
```

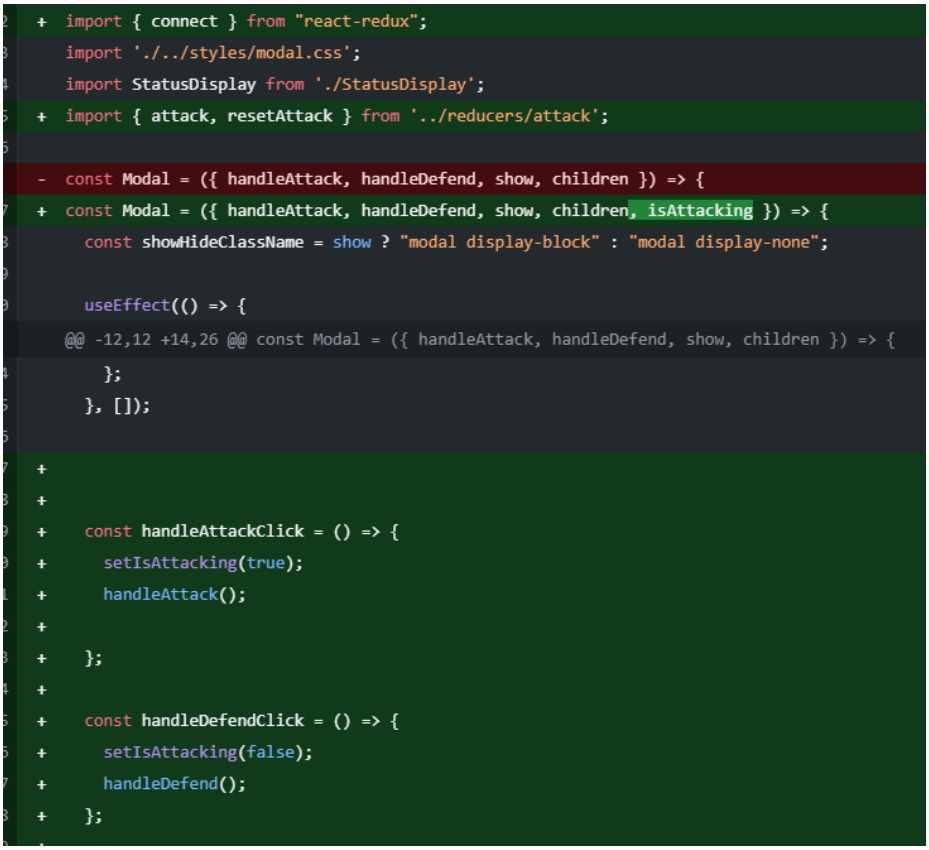
```
+ import React from "react";
+ import { connect } from "react-redux";
+ import playerImg from "./../assets/duck_up.gif";
+ import enemyImg from "./../assets/enemy_duck_up.gif";
+ import { showModal } from '../reducers/modalSlice';
+
+
+ const StatusDisplay = ({ playerHp, playerStrength, opponentHp, opponentStrength }) => {
+   return (
+     <div className="status-container">
+       <div className="status-box"><h2>Player Status</h2>
+       <p>HP: {playerHp}</p>
+       <p>Strength: {playerStrength}</p>
+       {showModal &&<img src={playerImg} ></img>}
+       </div>
+       =<div className="status-box"> <h2>Opponent Status</h2>
+       <p>HP: {opponentHp}</p>
+       <p>Strength: {opponentStrength}</p>
+       {showModal && <img src={enemyImg}></img>}
+       </div>
+     </div>
+   );
+ };
+
+ const mapStateToProps = (state) => ({
+   playerHp: state.playerController.player_hp,
+   playerStrength: state.playerController.player_strength,
+   opponentHp: state.opponent.opponent_hp,
+   opponentStrength: state.opponent.opponent_strength
+ });
+
+ export default connect(mapStateToProps)(StatusDisplay);
```

Changed Modal appereance:


```
+ import "./../styles/statusDisplay.css";

-
- const StatusDisplay = ({ playerHp, playerStrength, opponentHp, opponentStrength }) => {
+ const StatusDisplay = ({ playerHp, playerStrength, opponentHp, opponentStrength, show }) => {
    return (
      <div className="status-container">
-       <div className="status-box"><h2>Player Status</h2>
-       <p>HP: {playerHp}</p>
-       <p>Strength: {playerStrength}</p>
+       <div className="status-box">
        {showModal &&<img src={playerImg} ></img>}
+       <p className="hp">HP: {playerHp}</p>
+       <p className="strength">Strength: {playerStrength}</p>
        </div>
-       =<div className="status-box"> <h2>Opponent Status</h2>
-       <p>HP: {opponentHp}</p>
-       <p>Strength: {opponentStrength}</p>
-       {showModal && <img src={enemyImg}></img>}
+       =<div className="status-box">
+       {showModal && <img src={enemyImg}></img>}
+       <p  className="hp">HP: {opponentHp}</p>
+       <p className="strength">Strength: {opponentStrength}</p>
```

28/07/2024

Added green for player status:



```
+ import { connect } from "react-redux";
  import './../styles/modal.css';
  import StatusDisplay from './StatusDisplay';
+ import { attack, resetAttack } from '../reducers/attack';

- const Modal = ({ handleAttack, handleDefend, show, children }) => {
+ const Modal = ({ handleAttack, handleDefend, show, children, isAttacking }) => {
    const showHideClassName = show ? "modal display-block" : "modal display-none";

    useEffect(() => {
@@ -12,12 +14,26 @@ const Modal = ({ handleAttack, handleDefend, show, children }) => {
      };
    }, []);

+
+
+  const handleAttackClick = () => {
+    setIsAttacking(true);
+    handleAttack();
+
+  };
+
+  const handleDefendClick = () => {
+    setIsAttacking(false);
+    handleDefend();
+  };
```

```
+ const mapStateToProps = (state) => ({
+   isAttacking: state.isAttacking
+ });
+
+ const mapDispatchToProps = (dispatch) => ({
+   handleAttack: () => dispatch(attack()),
+   handleDefend: () => dispatch(resetAttack()),
+ });
+
+ export default connect(mapStateToProps, mapDispatchToProps)(Modal);
```

```
+  const StatusDisplay = ({ playerHp, playerStrength, opponentHp, opponentStrength, isAttacking }) => {
+
+    const playerStatusStyle = {
+      backgroundColor: isAttacking ? 'darkgreen' : 'white',
+    }
+
```

```
// Action Types
const ATTACK = 'attack/ATTACK';
const RESET_ATTACK = 'attack/RESET_ATTACK';

// Initial State
const initialState = {
  isAttacking: false,
};

// Reducer
const attackReducer = (state = initialState, action) => {
  switch (action.type) {
    case ATTACK:
      return { ...state, isAttacking: true };
    case RESET_ATTACK:
      return { ...state, isAttacking: false };
    default:
      return state;
  }
};

// Action Creators
export const attack = () => ({ type: ATTACK });
export const resetAttack = () => ({ type: RESET_ATTACK });

// Selectors
export const selectIsAttacking = (state) => state.attack.isAttacking;

export default attackReducer;
```

Added status timer:

```
+      if (show) {
+        preAttackTimer = setTimeout(() => {
+          setBackgroundColor('lightgreen');
+          attackTimer = setTimeout(() => {
+            setBackgroundColor('green');
+            setAttackPhase(true);
+          }, 2000); // Time between light green and full green
+        }, 2000); // Time before light green phase
+      }

+      return () => {
+        clearTimeout(attackTimer);
+        clearTimeout(preAttackTimer);
+        setBackgroundColor('white');
+        setAttackPhase(false);
+      };
+    }, [show, setBackgroundColor, setAttackPhase]);
```

29/07/2024

Added MapMatrix:

```
1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1
```

Changed status appereance and button position:

```
import React from "react";
import { connect } from "react-redux";
import playerImg from "./../assets/duck_up.gif";
import enemyImg from "./../assets/enemy_duck_up.gif";

const GameStatusDisplay = ({ playerHp, playerStrength, opponentHp, opponentStrength, isAttacking, statusBackgroundColor }) => {

  return (
    <div className="game-status-container">
      <div className="game-status-box" style={{ backgroundImage: "url('./../assets/marginImage.png')" }}>
        <p className="hp">HP: {playerHp}</p>
        <p className="strength">Strength: {playerStrength}</p>
      </div>
      <div className="game-status-box">
        <p className="hp">HP: {opponentHp}</p>
        <p className="strength">Strength: {opponentStrength}</p>
      </div>
    </div>
  );
};

const mapStateToProps = (state) => ({
  playerHp: state.playerController.player_hp,
  playerStrength: state.playerController.player_strength,
  opponentHp: state.opponent.opponent_hp,
  opponentStrength: state.opponent.opponent_strength,
  isAttacking: state.attack.isAttacking,
  statusBackgroundColor: state.attack.statusBackgroundColor,
});
```

```
+    display: flex;
+    flex-direction: row; /* Așea
+    height: 100vh; /* Asigură-te
+    padding: 0; /* Elimină paddi
+    box-sizing: border-box;
+  }
+
+
+  .map {
+    flex: 3; /* Ocupă mai mult s
     display: flex;
     flex-direction: column;
-    align-items: center;
-    justify-content: center;
-    padding-top: 200px;
+    align-items: center; /* Cent
+    justify-content: center; /*
+    z-index: 0;
+  }
+
+  .game-status {
+    flex: 1; /* Ocupă mai puțin
+    display: flex;
+    flex-direction: column;
+    align-items: flex-start; /*
+    justify-content: flex-start;
+    padding: 20px; /* Adaugă pad
     box-sizing: border-box;
   }
```

```
.status-container {
    display: flex;
    justify-content: space-around;
    width: 100%;
    padding: 20px;
    box-sizing: border-box;
    margin-bottom: 10px;
  }

  .status-box {
    background-color: rgba(255, 255, 255, 0.8);
    border: 1px solid black;
    border-radius: 10px;
    padding: 10px;
    width: 200px;
    text-align: center;
  }
```

Part of the logic of attack and defend buttons was added:

```
  playerHp: state.player.player_hp,
  playerStrength: state.player.player_strength,
  opponentHp: state.opponent.opponent_hp,
  opponentStrength: state.opponent.opponent_strength,
  isAttacking: state.attack.isAttacking,
  statusBackgroundColor: state.attack.statusBackgroundColor,
});
```

```
+    direction: state.direction,
+    isNpcMovable: state.isNpcMovable,
+    player_hp: state.player_hp,
+    player_strength: state.player_strength,
```

```
import { attack, resetAttack, setBackgroundColor, setAttackPhase
import { decreaseOpponentHealth } from './../reducers/opponent';
import { decreasePlayerHealth } from '../reducers/player';

const Modal = ({
  handleAttack,
  handleDefend,
  show,
  children,
  isAttacking,
  isAttackPhase,
  setBackgroundColor,
  setAttackPhase,
  setAttackTimerStarted,
  decreasePlayerHealth,
  decreaseOpponentHealth,
  setOpponentStatusColor,
  setTurn,
  currentTurn,
  opponentStrength
}) => {
  const [defendTimeout, setDefendTimeout] = useState(null);
```

```
// Randomly select whose turn it is
const isPlayerTurn = Math.random() > 0.5;
setTurn(isPlayerTurn ? 'player' : 'opponent');

// Set appropriate background colors
if (isPlayerTurn) {
  setBackgroundColor('lightgreen');
} else {
  setOpponentStatusColor('lightgreen');
}




attackTimer = setTimeout(() => {
  setBackgroundColor('green');
  if (isPlayerTurn) {
    console.log('player turn');
    setBackgroundColor('green');
  } else {
    console.log('opponent turn');
    setOpponentStatusColor('green');
    // Set defend timeout when opponent is green
    const timeoutId = setTimeout(() => {
      if (currentTurn === 'opponent') {
        setOpponentStatusColor('green');
        decreasePlayerHealth(Math.floor(Math.random() * opponentStrength) + 1);
        console.log("Player missed the defend window and took damage.");
      }
    }, 2000); // 2 seconds to defend
```

```
+ const SET_ATTACK_TIMER_STARTED = 'attack/SET_ATTACK_TIMER_STARTED';
+ const SET_OPPONENT_STATUS_COLOR = 'attack/SET_OPPONENT_STATUS_COLOR';
+ const SET_TURN = 'attack/SET_TURN';
+
+


  // Initial State
  const initialState = {
    isAttacking: false,
    statusBackgroundColor: 'white',
-   isAttackPhase: false
+   isAttackPhase: false,
+   attackTimerStarted: false,
+   opponentStatusColor: 'white',
+   currentTurn: 'player',
```

```
  case SET_ATTACK_TIMER_STARTED:
      return { ...state, attackTimerStarted: action.payload };
  case SET_OPPONENT_STATUS_COLOR:
      return { ...state, opponentStatusColor: action.payload };
  case SET_TURN:
      return { ...state, currentTurn: action.payload };
      default:
    return state;
 }
;
@ -32,5 +46,9 @@ export const attack = () => ({ type: ATTACK });
xport const resetAttack = () => ({ type: RESET_ATTACK });
xport const setBackgroundColor = (color) => ({ type: SET_BACKGROUND_COLOR, payload: color });
xport const setAttackPhase = (isPhase) => ({ type: SET_ATTACK_PHASE, payload: isPhase });
xport const setAttackTimerStarted = (started) => ({ type: SET_ATTACK_TIMER_STARTED, payload: started });
xport const setOpponentStatusColor = (color) => ({type: SET_OPPONENT_STATUS_COLOR, payload: color});
xport const setTurn = (turn) => ({ type: SET_TURN, payload: turn });
```

```
+ const DECREASE_PLAYER_HEALTH = "player/DECREASE_PLAYER_HEALTH"
+
+ // Initial state
+ const initialState = {
+   player_hp: 3,
+   player_strength: 10,
+ };
+
+ // Reducer
+ const playerReducer = (state = initialState, action) => {
+   switch (action.type) {
+     case SET_PLAYER_HP:
+         return {...state, player_hp: action.payload};
+     case SET_PLAYER_STRENGTH:
+         return {...state, player_strength: action.payload};
+     case DECREASE_PLAYER_HEALTH:
+         return { ...state, playerHealth: state.playerHealth - action.payload };
+     default:
+         return state;
+   }
+ };
+
+ // Action creators
+ export const moveOpponent = (x, y, direction) => ({
+   type: MOVE_OPPONENT,
+   payload: { x, y, direction }
+ });
+
+ export const decreasePlayerHealth = (amount) => ({ type: DECREASE_PLAYER_HEALTH, payload: amount });
+ export const setPlayerHp = (hp) => ({ type: SET_PLAYER_HP, payload: hp });
+ export const setPlayerStrength = (strength) => ({ type: SET_PLAYER_STRENGTH, payload: strength });
+
+ export default playerReducer;
```

Added map based on matrix:

```
<tbody>
  {mapMatrix.map((row, rowIndex) => (
    <tr key={rowIndex}>
      {row.map((cell, colIndex) => {
        const isBorderCell = cell === 1;
        let className = "map-cell";

        if (colIndex === x && rowIndex === y) {
          className = "duck-cell";
        } else if (colIndex === opponent.x && rowIndex === opponent.y) {
          className = "opponent-cell";
        }

        return (
          <td
            key={`${rowIndex}-${colIndex}`}
            className={className}
            style={isBorderCell ? { backgroundImage: `url(${marginImage})`, backgroundSize: 'cover' } : {}}
          >
            {className === "duck-cell" && (
              <img
                src={getDuckImage(direction, true)}
                alt="Duck"
                className="duck-image"
              />
            )}
            {className === "opponent-cell" && (
              <img
                src={getDuckImage(opponent.direction, false)}
                alt="Opponent Duck"
                className="duck-image"
              />
            )}
```

Added innaccesibility to solid cells:

```
+       if (state.isNpcMovable) {
+         newY = Math.max(state.y - 1, 1);
+         if (mapMatrix[newY][state.x] !== 1) {
+           return { ...state, y: newY, direction: "UP" };
+         }
+       }
+       return state;
+
      case MOVE_DOWN:
-       if (!state.isNpcMovable) return state; // Nu permite mișcarea dacă NPC
-       return { ...state, y: Math.min(state.y + 1, 8), direction: "DOWN" };
+       if (state.isNpcMovable) {
+         newY = Math.min(state.y + 1, 8);
+         if (mapMatrix[newY][state.x] !== 1) {
+           return { ...state, y: newY, direction: "DOWN" };
+         }
+       }
+       return state;
+
      case MOVE_LEFT:
-       if (!state.isNpcMovable) return state; // Nu permite mișcarea dacă NPC
-       return { ...state, x: Math.max(state.x - 1, 1), direction: "LEFT" };
+       if (state.isNpcMovable) {
+         newX = Math.max(state.x - 1, 1);
+         if (mapMatrix[state.y][newX] !== 1) {
+           return { ...state, x: newX, direction: "LEFT" };
+         }
+       }
```

```
+       if (state.isNpcMovable) {
+         newX = Math.min(state.x + 1, 8);
+         if (mapMatrix[state.y][newX] !== 1) {
+           return { ...state, x: newX, direction: "RIGHT" };
+         }
+       }
+       return state;
+
      case SET_NPC_MOVABLE:
        return { ...state, isNpcMovable: action.payload };
-    default:
+
+     default:
        return state;
    }
  };

- // Action Creators (creatori de acțiuni)
+ // Action Creators
  export const moveUp = () => ({ type: MOVE_UP });
  export const moveDown = () => ({ type: MOVE_DOWN });
- export const moveLeft = () => ({ type: MOVE_LEFT});
- export const moveRight = () => ({ type: MOVE_RIGHT});
- export const setNpcMovable = (isMovable) => ({ type: SET_NPC_MOVABLE, payload: isMovable});
-
+ export const moveLeft = () => ({ type: MOVE_LEFT });
+ export const moveRight = () => ({ type: MOVE_RIGHT });
+ export const setNpcMovable = (isMovable) => ({ type: SET_NPC_MOVABLE, payload: isMovable });
```