Detect overlapping circles

Problem Statement -

Draw circles on the screen on the click and whenever two circles overlap change the color of the second circle.

- When a user clicks anywhere on the DOM, create a circle around it of a radius of 100px with a red background.
- If two or more circles overlap, change the background of the later circle.

Let us understand the logic of creating the circle first.

As we have to create the circle with a radius of 100px (200px diameter), rather than generating the circle on the click, we will store the coordinates of the position where the circle should be generated when the user clicks and then create circles out of these coordinates.

As all the circles will be in absolute position so that they can be freely placed on the screen, we will calculate the top, bottom, left, and right positions that will help in placement as well as detecting if two circles are colliding.

Get the clientX and clientY coordinates when the user clicks and align the circle around with a simple calculation so that it is placed in the center. Also before updating the state check if the current circle is overlapping with the existing circles then update the background color of the current.

```
// helper function to gather configuration when user clicks
const draw = (e) => {
  // get the coordinates where user has clicked
  const { clientX, clientY } = e;

  // decide the position where circle will be created and placed
```

```
  // as the circle is of 100 radius (200 diameter), we are subtracting the
values
  // so that circle is placed in the center
  // set the initial background color to red
  setElementsCoordinates((prevState) => {
    const current = {
      top: clientY - 100,
      left: clientX - 100,
      right: clientX - 100 + 200,
      bottom: clientY - 100 + 200,
      background: "red",
    };

    // before making the new entry
    // check with the existing circles
    for (let i = 0; i < prevState.length; i++) {
      // if the current circle is colliding with any existing
      // update the background color of the current
      if (elementsOverlap(current, prevState[i])) {
        current.background = getRandomColor();
        break;
      }
    }

    return [...prevState, current];
  });
};
```

Assign the event listener and draw the circle on the click.

```
// assign the click event
  useEffect(() => {
    document.addEventListener("click", draw);
    return () => {
      document.removeEventListener("click", draw);
    };
  }, []);
```

Helper function to detect collision and generate random colors.

```
// helper function to generate a random color
const getRandomColor = () => {
  const letters = "0123456789ABCDEF";
  let color = "#";
  for (let i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
};

// helper function to detect if two elements are overlapping
const elementsOverlap = (rect1, rect2) => {
  const collide = !(
    rect1.top > rect2.bottom ||
    rect1.right < rect2.left ||
    rect1.bottom < rect2.top ||
    rect1.left > rect2.right
  );

  return collide;
};
```

Generate the circles from the coordinates which we have stored after the user has clicked. As the detection is done before making entry into the state, the circles are generated with different colors if they collide.

```
// circle element
const Circle = ({ top, left, background }) => {
  return (
    <div
      style={{
        position: "absolute",
        width: "200px",
        height: "200px",
        borderRadius: "50%",
        opacity: "0.5",
        background,
        top,
```

```
        left,
      }}
    ></div>
  );
};

return (
  <div>
    {/* render each circle */}
    {elementsCoordinates.map((e) => (
      <Circle {...e} key={e.top + e.left + e.right} />
    ))}
  </div>
);
```

Putting everything together.

```
import { useEffect, useState } from "react";

// helper function to generate a random color
const getRandomColor = () => {
  const letters = "0123456789ABCDEF";
  let color = "#";
  for (let i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
};

// helper function to detect if two elements are overlapping
const elementsOverlap = (rect1, rect2) => {
  const collide = !(
    rect1.top > rect2.bottom ||
    rect1.right < rect2.left ||
    rect1.bottom < rect2.top ||
    rect1.left > rect2.right
  );

  return collide;
};
```

```
const Example = () => {
  // store the configuration of each circle
  const [elementsCoordinates, setElementsCoordinates] = useState([]);

  // helper function to gather configuration when user clicks
  const draw = (e) => {
    // get the coordinates where user has clicked
    const { clientX, clientY } = e;

    // decide the position where circle will be created and placed
    // as the circle is of 100 radius (200 diameter), we are subtracting
the values
    // so that circle is placed in the center
    // set the initial background color to red
    setElementsCoordinates((prevState) => {
      const current = {
        top: clientY - 100,
        left: clientX - 100,
        right: clientX - 100 + 200,
        bottom: clientY - 100 + 200,
        background: "red",
      };

      // before making the new entry
      // check with the existing circles
      for (let i = 0; i < prevState.length; i++) {
        // if the current circle is colliding with any existing
        // update the background color of the current
        if (elementsOverlap(current, prevState[i])) {
          current.background = getRandomColor();
          break;
        }
      }

      return [...prevState, current];
    });
  };

  // assign the click event
  useEffect(() => {
    document.addEventListener("click", draw);
    return () => {
```

```
      document.removeEventListener("click", draw);
    };
  }, []);

  // circle element
  const Circle = ({ top, left, background }) => {
    return (
      <div
        style={{
          position: "absolute",
          width: "200px",
          height: "200px",
          borderRadius: "50%",
          opacity: "0.5",
          background,
          top,
          left,
        }}
      ></div>
    );
  };

  return (
    <div>
      {/* render each circle */}
      {elementsCoordinates.map((e) => (
        <Circle {...e} key={e.top + e.left + e.right} />
      ))}
    </div>
  );
};

export default Example;
```

Output