```
const config = {
  SCREEN_SIZE: 800,
  ELEMENT_SIZE: 100,
  MIDDLE: 800/2 - 100/2,
  MOVEMENT: 5
}

class Bullet extends React.Component {
  render() {
    // console.log('Bullet:render')
    const style = {
      width: config.ELEMENT_SIZE,
      height: config.ELEMENT_SIZE,
      background: 'pink',
      position: 'absolute',
      bottom: this.props.bulletPos,
      left: this.props.playerPos
    }
    return <div style={style} />
  }
}

class Enemy extends React.Component {
  constructor(props) {
    super(props)

    this.state = {
      display: 'block',
      left: props.left
    }

    this.destroyed = false
  }

  componentWillMount() {
    let movement = config.MOVEMENT

    const frame = () => {
      if (this.state.left + config.ELEMENT_SIZE == config.SCREEN_SIZE ||
this.state.left === 0) {
        movement *= -1
      }
      this.setState({
        left: this.state.left + movement
      })

      const me = {
        min: {
          X: this.state.left,
          Y: this.props.bottom - config.ELEMENT_SIZE
        },
        max: {
          X: this.state.left + config.ELEMENT_SIZE,
          Y: this.props.bottom
        }
      }

      const other = {
        min: {
          X: this.props.bulletPos.x,
          Y: this.props.bulletPos.y - config.ELEMENT_SIZE
        },
        max: {
          X: this.props.bulletPos.x + config.ELEMENT_SIZE,
          Y: this.props.bulletPos.y
        }
      }

      // colision detection
      if(!(
        me.max.X < other.min.X ||
        me.max.Y < other.min.Y ||
        me.min.X > other.max.X ||
        me.min.Y > other.max.Y
      )) {
        this.handleDestroy()
      }

      requestAnimationFrame(frame)
    }

    requestAnimationFrame(frame)
  }

  handleDestroy() {
    if (!this.destroyed) {
      this.destroyed = true
      console.log('Enemy destroyed')

      this.setState({
        display: 'none'
      })

      this.props.onEnemyDestroyed()
    }
  }

  render() {
    // console.log('Enemy:render')
    const style = {
      width: config.ELEMENT_SIZE,
      height: config.ELEMENT_SIZE,
      background: 'yellow',
      position: 'absolute',
      bottom: this.props.bottom,
      left: this.state.left,
      display: this.state.display
    }
    return <div style={style} />
  }
}


class Player extends React.Component {
  render() {
    // console.log('Player:render')
    const style = {
      width: config.ELEMENT_SIZE,
      height: config.ELEMENT_SIZE,
      background: 'blue',
```

```
      position: 'absolute',
      bottom: 0,
      left: this.props.position
    }
    return <div style={style} />
  }
}

class Game extends React.Component {
  constructor() {
    super()

    this.state = {
      playerPos: config.MIDDLE,
      bulletPos: 1100,
      enemies: 6
    }

    this.onEnemyDestroyed = this.onEnemyDestroyed.bind(this)
  }

  shoot() {
    this.setState({
      bulletPos: config.ELEMENT_SIZE
    })
    const frame = () => {
      this.setState({
        bulletPos: this.state.bulletPos + config.MOVEMENT
      })
      requestAnimationFrame(frame)
    }
    requestAnimationFrame(frame)
  }

  onEnemyDestroyed() {
    this.setState({
      enemies: this.state.enemies - 1
    })
    if (this.state.enemies - 1 <= 0) {
      alert('you win!')
    }
  }

  handleKeyDown(e) {
    if (e.key === "ArrowRight") {
      this.setState({
        playerPos: this.state.playerPos + config.MOVEMENT
      })
    }
    if (e.key === "ArrowLeft") {
      this.setState({
        playerPos: this.state.playerPos - config.MOVEMENT
      })
    }
    if (e.key === " ") {
      this.shoot()
    }
  }

  render() {
    const style = {
      width: config.SCREEN_SIZE,
      height: config.SCREEN_SIZE,
      background: 'red',
      position: 'relative'
    }

    const bulletPos = { y: this.state.bulletPos, x: this.state.playerPos }

    return (
      <div style={style} onKeyDown={this.handleKeyDown.bind(this)}
tabIndex="0">
        <Enemy
          bulletPos={bulletPos}
          bottom={config.MIDDLE} left={150}
          onEnemyDestroyed={this.onEnemyDestroyed}
        />
        <Enemy
          bulletPos={bulletPos}
          bottom={config.MIDDLE} left={350}
          onEnemyDestroyed={this.onEnemyDestroyed}
        />
        <Enemy
          bulletPos={bulletPos}
          bottom={config.MIDDLE} left={550}
           onEnemyDestroyed={this.onEnemyDestroyed}
        />
        <Enemy
          bulletPos={bulletPos}
          bottom={config.MIDDLE + 150} left={150}
          onEnemyDestroyed={this.onEnemyDestroyed}
        />
        <Enemy
          bulletPos={bulletPos}
          bottom={config.MIDDLE + 150} left={350}
          onEnemyDestroyed={this.onEnemyDestroyed}
        />
        <Enemy
          bulletPos={bulletPos}
          bottom={config.MIDDLE + 150} left={550}
          onEnemyDestroyed={this.onEnemyDestroyed}
        />
        <Bullet bulletPos={this.state.bulletPos} playerPos=
{this.state.playerPos}  />
        <Player position={this.state.playerPos} />
      </div>
    )
  }
}


ReactDOM.render(
  <Game />,
  document.getElementById('reactroot')
)
```