Cmpe 496 HW1 Drawing Editor - Object Oriented Drawing Editor with Typescript

* This project is an implementation of drawing editor using object oriented paradigm.
* This program is written with vue + typescript. It uses classes to represent objects.

**Requirements**

* There shall be three shapes in this program: Circle, rectangle, line.
* Users shall be able to move circles, rectangles.
* Users shall be able to draw all the shapes.
* Users shall be able to delete all circles.
* Users shall be able to delete all rectangles.
* Users shall be able to delete all lines.
* Users shall be able to change background color of the canvas.
* Users shall be able to change width and height of the canvas.
* The shapes shall stay in the canvas after moving them.
* Shapes shall be able to collide each other.
* There shall be a class Shape that has
  + name of the shape - string
  + draw() method that draws the shape into canvas.
* Line class shall have
  + x1 - real number
  + x2 - real number
  + y1 - real number
  + y2 - real number
* Rectangle class shall extend Shape class and shall have
  + x1 - real number
  + y1 - real number
  + x2 - real number
  + y2 - real number
* There shall be a Circle class that has
  + centerX - real number
  + centerY - real number
  + radius - real number
* Canvas class shall have
  + width - integer
  + height - integer
  + backgroundColor - html hls color value as string

**Stages of Development**

First, I declared my program’s requirements as written in above statements.

Second, I created Canvas component in vue and gave it the attributes that are defined above. I gave a mouse click event listener to the canvas that I define in main component. This function listens the clicked position and dependent on the operation mode of the canvas something happens. This function is my main function.

Third, I created several operation modes. For each shape there is an operation mode for the creation of the shape. The descriptions of how to add a shape is shown when hovering over the button associated with a shape.

Fourth, I added shape creation codes. For example, in order to add a circle, two points are chosen, the center point and a point on the border.

Fifth, I added the drag functionality by adding mouse up and mouse move event listeners to all shapes.

Sixth, I added canvas background color and size changer widget. I made the widget hidable so that users are not distracted with the widget after their operation is done.

**Conclusion**

I learned a lot of things in this project.

* Designing user interfaces object oriented is a good idea since it saves you code.
* Making user interface simple and having tutorials is important.
* Making menus hidable is also a good idea to keep user inside of the environment.

**Code**

**Canvas.vue**

<template>

  <div

    id="canvas"

    @click="this.$props.mouseClick"

    :style="{

      width: this.$props.width + 'px',

      height: this.$props.height + 'px',

      backgroundColor: this.$props.backgroundColor

    }"

  >

    <slot></slot>

  </div>

</template>

<script lang="ts">

import Vue from "vue";

export default Vue.extend({

  props: {

    width: Number,

    height: Number,

    backgroundColor: String,

    mouseClick: Function

  },

  data: function() {

    return {};

  },

  methods: {}

});

</script>

<style></style>

**Circle.vue**

<template>

  <span

    @mousedown="dragMouseDown"

    :style="{

      height: this.$props.radius + 'px',

      width: this.$props.radius + 'px',

      top: this.top + 'px',

      left: this.left + 'px',

      backgroundColor: 'transparent',

      borderRadius: '50%',

      border: '2px solid black',

      zIndex: 100,

      position: 'absolute',

      display: 'inline-block',

      cursor: 'move'

    }"

  ></span>

</template>

<script>

import Vue from "vue";

import Component from "vue-class-component";

@Component({

  props: {

    centerX: Number,

    centerY: Number,

    radius: Number,

    draggable: Boolean,

    canvasWidth: Number,

    canvasHeight: Number

  },

  data: function() {

    return {

      pos1: 0,

      pos2: 0,

      pos3: 0,

      pos4: 0,

      top: this.$props.centerY,

      left: this.$props.centerX

    };

  }

})

export default class CircleComponent extends Vue {

  constructor() {

    super();

  }

  closeDragElement() {

    if (this.$props.draggable) {

      // stop moving when mouse button is released:

      this.window.document.onmouseup = null;

      this.window.document.onmousemove = null;

    }

  }

  elementDrag(e) {

    e = e || this.window.event;

    e.preventDefault();

    if (this.$props.draggable) {

      // calculate the new cursor position:

      this.pos1 = this.pos3 - e.clientX;

      this.pos2 = this.pos4 - e.clientY;

      this.pos3 = e.clientX;

      this.pos4 = e.clientY;

      // set the element's new position:

      let newX = this.$el.offsetTop - this.pos2;

      let newY = this.$el.offsetLeft - this.pos1;

      if (newX < 0) newX = 0;

      if (newY < 0) newY = 0;

      if (newX + this.$props.radius > this.$props.canvasHeight)

        newX = this.$props.canvasHeight - this.$props.radius;

      if (newY + this.$props.radius > this.$props.canvasWidth)

        newY = this.$props.canvasWidth - this.$props.radius;

      this.top = newX;

      this.left = newY;

    }

  }

  dragMouseDown(e) {

    e = e || this.window.event;

    e.preventDefault();

    if (this.$props.draggable) {

      // get the mouse cursor position at startup:

      this.pos3 = e.clientX;

      this.pos4 = e.clientY;

      this.window.document.onmouseup = this.closeDragElement;

      // call a function whenever the cursor moves:

      this.window.document.onmousemove = this.elementDrag;

    }

  }

}

</script>

**Line.vue**

<template>

  <div id="line">

    <svg

      :style="{

        height: Math.abs(this.$props.y1 - this.$props.y2) + 'px',

        width: Math.abs(this.$props.x1 - this.$props.x2) + 'px',

        top: this.top + 'px',

        left: this.left + 'px',

        backgroundColor: 'transparent',

        zIndex: 99,

        position: 'absolute',

        display: 'inline-block'

      }"

    >

      <line

        :x1="this.$props.x1 - this.left"

        :y1="this.$props.y1 - this.top"

        :x2="this.$props.x2 - this.left"

        :y2="this.$props.y2 - this.top"

        style="stroke:rgb(255,0,0);stroke-width:2"

      />

    </svg>

  </div>

</template>

<script>

import Vue from "vue";

import Component from "vue-class-component";

@Component({

  props: {

    x1: Number,

    y1: Number,

    x2: Number,

    y2: Number,

    canvasWidth: Number,

    canvasHeight: Number

  },

  data: function() {

    return {

      pos1: 0,

      pos2: 0,

      pos3: 0,

      pos4: 0,

      top: Math.min(this.$props.y1, this.$props.y2),

      left: Math.min(this.$props.x1, this.$props.x2)

    };

  }

})

export default class LineComponent extends Vue {

  constructor() {

    super();

  }

}

</script>

**Rectangle.vue**

<template>

  <span

    @mousedown="dragMouseDown"

    :style="{

      height: this.$props.height + 'px',

      width: this.$props.width + 'px',

      top: this.top + 'px',

      left: this.left + 'px',

      backgroundColor: 'transparent',

      border: '2px solid black',

      zIndex: 100,

      position: 'absolute',

      display: 'inline-block',

      cursor: 'move'

    }"

  ></span>

</template>

<script>

import Vue from "vue";

import Component from "vue-class-component";

@Component({

  props: {

    x: Number,

    y: Number,

    height: Number,

    width: Number,

    draggable: Boolean,

    canvasWidth: Number,

    canvasHeight: Number

  },

  data: function() {

    return {

      pos1: 0,

      pos2: 0,

      pos3: 0,

      pos4: 0,

      top: this.$props.y,

      left: this.$props.x

    };

  }

})

export default class Rectangle extends Vue {

  // Initial data can be declared as instance properties

  // message: string = "Hello!";

  // Component methods can be declared as instance methods

  // onClick(): void {

  //   window.alert(this.message);

  // }

  constructor() {

    super();

  }

  closeDragElement() {

    if (this.$props.draggable) {

      // stop moving when mouse button is released:

      this.window.document.onmouseup = null;

      this.window.document.onmousemove = null;

    }

  }

  elementDrag(e) {

    e = e || this.window.event;

    e.preventDefault();

    if (this.$props.draggable) {

      // calculate the new cursor position:

      this.pos1 = this.pos3 - e.clientX;

      this.pos2 = this.pos4 - e.clientY;

      this.pos3 = e.clientX;

      this.pos4 = e.clientY;

      // set the element's new position:

      let newX = this.$el.offsetTop - this.pos2;

      let newY = this.$el.offsetLeft - this.pos1;

      if (newX < 0) newX = 0;

      if (newY < 0) newY = 0;

      if (newX + this.$props.height > this.$props.canvasHeight)

        newX = this.$props.canvasHeight - this.$props.height;

      if (newY + this.$props.width > this.$props.canvasWidth)

        newY = this.$props.canvasWidth - this.$props.width;

      this.top = newX;

      this.left = newY;

    }

  }

  dragMouseDown(e) {

    e = e || this.window.event;

    e.preventDefault();

    if (this.$props.draggable) {

      // get the mouse cursor position at startup:

      this.pos3 = e.clientX;

      this.pos4 = e.clientY;

      this.window.document.onmouseup = this.closeDragElement;

      // call a function whenever the cursor moves:

      this.window.document.onmousemove = this.elementDrag;

    }

  }

}

</script>

**App.vue**

<template>

  <div id="app">

    <b-container>

      <b-row>

        <!-- <Line v-for="line in lines" :key="line.id"> </Line>

        <Rectangle v-for="rectangle in rectangles" :key="rectangle.id">

        </Rectangle> -->

      </b-row>

      <b-row align-h="center">

        <b-button v-b-toggle.collapse-1 variant="primary">{{

          visible ? "Hide Canvas Settings" : "Show Canvas Settings"

        }}</b-button>

      </b-row>

      <b-row align-h="center">

        <b-collapse

          style="zIndex:1; background-color: darkblue;"

          id="collapse-1"

          class="mt-2"

          v-model="visible"

        >

          <b-form>

            <label class="white" for="width">Width</label>

            <b-form-input

              id="width"

              v-model="width"

              type="range"

              min="100"

              :max="window.innerWidth"

            ></b-form-input>

            <p class="white">Width: {{ width }}</p>

            <label class="white" for="height">Height</label>

            <b-form-input

              id="height"

              v-model="height"

              type="range"

              min="100"

              :max="window.innerHeight"

            ></b-form-input>

            <p class="white">Height: {{ height }}</p>

          </b-form>

          <p class="white">Select Color:</p>

          <verte

            picker="wheel"

            model="rgb"

            value="#00f"

            v-model="backgroundColor"

          ></verte>

        </b-collapse>

      </b-row>

      <b-row align-h="center">

        <p style="color: white;">

          Current Operation:

          {{ currentSelection ? "Add " + currentSelection : "Drag" }}

        </p>

      </b-row>

      <b-row align-h="center">

        <b-button-group class="menu">

          <b-button

            @click="onCircleSelect"

            v-b-tooltip.hover

            title="Draw a circle by first clicking on center point, then a point that will be on the border."

            >Circle</b-button

          >

          <b-button

            @click="onLineSelect"

            v-b-tooltip.hover

            title="Draw a line by clicking to two points"

            >Line</b-button

          >

          <b-button

            v-b-tooltip.hover

            @click="onRectSelect"

            title="Draw a rectangle or square by clicking on two points that are on the diagonal."

            >Rectangle/Square</b-button

          >

          <b-button

            v-b-tooltip.hover

            @click="onDeselect"

            title="Press this if you do not want to add the shape you selected."

            >Drag</b-button

          >

        </b-button-group>

      </b-row>

      <b-row align-h="center">

        <b-button-group class="otherMenu">

          <b-button @click="onDeleteCircles">Delete Circles</b-button>

          <b-button @click="onDeleteRectangles">Delete Rectangles</b-button>

          <b-button @click="onDeleteLines">Delete Lines</b-button>

        </b-button-group>

      </b-row>

      <b-row align-h="center">

        <Canvas

          id="canvas"

          :mouseClick="onMouseClick"

          :width="+width"

          :height="+height"

          :backgroundColor="backgroundColor"

        >

          <CircleComponent

            v-for="(circle, index) in circles"

            :key="index + circle.id"

            :radius="circle.radius"

            :centerX="circle.x"

            :centerY="circle.y"

            :draggable="currentSelection == ''"

            :canvasWidth="+width"

            :canvasHeight="+height"

          />

          <Rectangle

            v-for="(rect, index) in rectangles"

            :key="index + rect.id"

            :width="rect.width"

            :height="rect.height"

            :x="rect.x"

            :y="rect.y"

            :draggable="currentSelection == ''"

            :canvasWidth="+width"

            :canvasHeight="+height"

          />

          <LineComponent

            v-for="(line, index) in lines"

            :key="index + line.id"

            :x1="line.x1"

            :y1="line.y1"

            :x2="line.x2"

            :y2="line.y2"

            :canvasWidth="+width"

            :canvasHeight="+height"

          />

        </Canvas>

      </b-row>

    </b-container>

  </div>

</template>

<script>

import Vue from "vue";

import Canvas from "./components/Canvas.vue";

import CircleComponent from "./components/Circle.vue";

import Rectangle from "./components/Rectangle.vue";

import LineComponent from "./components/Line.vue";

// eslint-disable-next-line @typescript-eslint/no-var-requires

export default Vue.extend({

  data: function() {

    return {

      width: 500,

      height: 500,

      backgroundColor: "#00f",

      visible: false,

      currentSelection: "",

      firstPointX: 0,

      firstPointY: 0,

      selectingFirst: false,

      selecting: false,

      circles: [],

      rectangles: [],

      lines: []

    };

  },

  components: {

    Canvas,

    CircleComponent,

    Rectangle,

    LineComponent

  },

  methods: {

    onCircleSelect() {

      this.currentSelection = "Circle";

      this.selectingFirst = true;

      this.selecting = true;

      this.firstPointX = 0;

      this.firstPointY = 0;

    },

    onRectSelect() {

      this.currentSelection = "Rectangle/Square";

      this.selectingFirst = true;

      this.selecting = true;

      this.firstPointX = 0;

      this.firstPointY = 0;

    },

    onLineSelect() {

      this.currentSelection = "Line";

      this.selectingFirst = true;

      this.selecting = true;

      this.firstPointX = 0;

      this.firstPointY = 0;

    },

    onDeselect() {

      this.currentSelection = "";

      this.selectingFirst = false;

      this.selecting = false;

      this.firstPointX = 0;

      this.firstPointY = 0;

    },

    onMouseClick(event) {

      const canvas = document.querySelector("#canvas");

      let x, y;

      if (canvas != null) {

        x = event.pageX - canvas.offsetLeft;

        y = event.pageY - canvas.offsetTop;

        if (this.currentSelection == "Circle") {

          if (this.selecting) {

            if (this.selectingFirst == true) {

              this.firstPointX = x;

              this.firstPointY = y;

              this.selectingFirst = false;

            } else {

              const circle = {

                x: x >= this.firstPointX ? this.firstPointX : x,

                y: y >= this.firstPointY ? this.firstPointY : y,

                radius: Math.sqrt(

                  (this.firstPointX - x) \* (this.firstPointX - x) +

                    (this.firstPointY - y) \* (this.firstPointY - y)

                ),

                id: new Date().toLocaleString("en-us")

              };

              this.circles.push(circle);

              this.selectingFirst = true;

            }

          }

        } else if (this.currentSelection == "Rectangle/Square") {

          if (this.selecting) {

            if (this.selectingFirst == true) {

              this.firstPointX = x;

              this.firstPointY = y;

              this.selectingFirst = false;

            } else {

              const rectangle = {

                x: Math.min(x, this.firstPointX),

                y: Math.min(y, this.firstPointY),

                width: Math.abs(x - this.firstPointX),

                height: Math.abs(y - this.firstPointY),

                id: new Date().toLocaleString("en-us")

              };

              this.rectangles.push(rectangle);

              this.selectingFirst = true;

            }

          }

        } else if (this.currentSelection == "Line") {

          if (this.selecting) {

            if (this.selectingFirst == true) {

              this.firstPointX = x;

              this.firstPointY = y;

              this.selectingFirst = false;

            } else {

              const line = {

                x1: x,

                y1: y,

                x2: this.firstPointX,

                y2: this.firstPointY,

                id: new Date().toLocaleString("en-us")

              };

              this.lines.push(line);

              this.selectingFirst = true;

            }

          }

        }

      }

    },

    onDeleteCircles() {

      this.circles = [];

    },

    onDeleteRectangles() {

      this.rectangles = [];

    },

    onDeleteLines() {

      this.lines = [];

    }

  }

});

</script>

<style lang="scss">

#app {

  font-family: Avenir, Helvetica, Arial, sans-serif;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  text-align: center;

  color: #2c3e50;

  margin-top: 60px;

  display: flex;

  flex-direction: column;

  align-items: center;

}

.white {

  color: white;

}

#canvas {

  position: absolute;

  top: 12.5rem;

  z-index: 0;

}

.verte\_\_icon {

  border: 1px solid white;

}

.menu {

  position: absolute !important;

  top: 7.5rem;

  z-index: 0;

}

.otherMenu {

  position: absolute !important;

  top: 10rem;

  z-index: 0;

}

</style>

**Main.ts**

import Vue from "vue";

import App from "./App.vue";

import BootstrapVue from "bootstrap-vue";

import "../public/styles/global.scss";

import "verte/dist/verte.css";

// eslint-disable-next-line @typescript-eslint/no-var-requires

import Verte from "verte";

Vue.config.productionTip = false;

Vue.use(BootstrapVue);

Vue.component(Verte.name, Verte);

Vue.prototype.window = window;

// register component globally

new Vue({

  render: h => h(App)

}).$mount("#app");

**Thanks**

**Serkan Özel**

**2015400123**

**serkan.ozel@boun.edu.tr**