Uniwersytet Bielsko-Bialski

Sprawozdanie

Zajęcia: Grafika Komputerowa
(Ćwiczenia laboratoryjne)
Prowadzący: prof dr. hab. Vasyl Martsenyuk

Laboratoria nr: 3

Temat ćwiczenia:

Zadanie_hierarchia

Maksymilian Wójcik
Imformatyka I stopnia
niestacjonarne
4 semestr
gr. 1A

1.Polecenie:

Opracować scenę hierarchiczną zgodnie z obrazem używając zamiast kół wielokąty obracające się (animacja!) według wariantu. Opracowanie powinno być w jednym z języków: Java lub JavaScript,

na dwa sposoby:

(a) używając hierarchiję funkcje (sposób subroutinowy)

(b) tworząc graf sceny (sposób obiektowy). W tym celu proponuję do pobrania odpowiedni pliki

2. Wykorzystane komendy:

3. Wynik działania:

a)

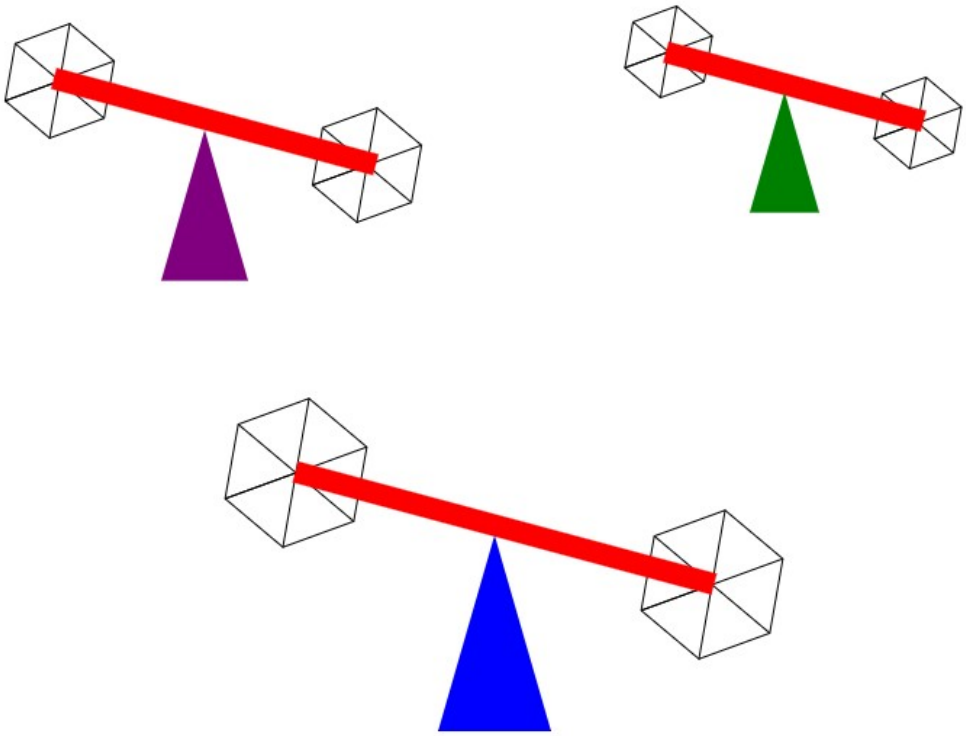
```
class Shape {
  constructor(centerX, centerY, size, sides, color = "#000") {
    this.centerX = centerX;
    this.centerY = centerY;
    this.size = size;
    this.sides = sides;
    this.color = color;
  }

  draw(ctx, angle) {
    ctx.beginPath();
    ctx.moveTo(
      this.centerX + this.size * Math.cos(angle),
      this.centerY + this.size * Math.sin(angle)
    );

    for (let i = 1; i <= this.sides; i++) {
      angle += (2 * Math.PI) / this.sides;
      const x = this.centerX + this.size * Math.cos(angle);
      const y = this.centerY + this.size * Math.sin(angle);
      ctx.lineTo(x, y);
      ctx.moveTo(this.centerX, this.centerY);
      ctx.lineTo(x, y);
    }

    ctx.closePath();
    ctx.strokeStyle = this.color;
    ctx.stroke();
  }
}
```

b)

```
function renderPolygon(cx, cy, r, sides) {
  context.beginPath();
  context.moveTo(
    cx + r * Math.cos(rotationAngle),
    cy + r * Math.sin(rotationAngle)
  );

  for (let i = 1; i <= sides; i++) {
    rotationAngle += (2 * Math.PI) / sides;
    const x = cx + r * Math.cos(rotationAngle);
    const y = cy + r * Math.sin(rotationAngle);
    context.lineTo(x, y);
    context.moveTo(cx, cy);
    context.lineTo(x, y);
  }

  context.closePath();
  context.strokeStyle = "#000";
  context.stroke();
}

function renderRotatedRect(x, y, w, h, angle, color) {
  context.save();
  context.translate(x, y);
  context.rotate((angle * Math.PI) / 180);
  context.fillStyle = color;
  context.fillRect(-w / 2, -h / 2, w, h);
  context.restore();
}

function renderTriangle(cx, cy, size, color) {
  const x1 = cx - size / 2;
  const y1 = cy + size / Math.sqrt(3);
  const x2 = cx + size / 2;
  const y2 = cy + size / Math.sqrt(3);
  const x3 = cx;
  const y3 = cy - (2 * size) / Math.sqrt(3);
```

4.Wynik działania:

a i b)



5.Wnioski

Sposób subroutinowy:

-Użycie hierarchii funkcji jest prostsze w implementacji, ale trudniejsze do zarządzania, gdy scena staje się bardziej złożona.

-Wszystkie przekształcenia są zarządzane ręcznie, co może prowadzić do bardziej skomplikowanego kodu.

Sposób obiektowy:

-Użycie grafu sceny umożliwia łatwiejsze zarządzanie złożonymi hierarchiami i przekształceniami.

-Każdy węzeł może mieć własne dzieci, co ułatwia dodawanie i modyfikowanie sceny.

-Kod jest bardziej modularny i zrozumiały.

Oba podejścia mają swoje zalety i wady, a wybór metody zależy od specyfiki projektu oraz osobistych preferencji programisty.

Kod źródłowy: https://github.com/mwojcik123/UBB-GK-MW