SPRAWOZDANIE

Zajęcia: Grafika komputerowa
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium: Grafika Komputerowa
28.04.2020
Temat: „Tekstury w OpenGL”
Wariant: -

Michał Krzyżowski
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.1b

1. Polecenie:

W obecnych laboratoriach należało, zteksturować piramidę, poprzez użycie tekstur z bufora kolorów oraz tekstury z pliku (ziemia, chmura, cegły), do tego należało opracować w uprzednio przygotowanym programie 2 metody textureFromPainting() oraz textureFromResource().

2. Wprowadzane dane:

Jeżeli mowa o wprowadzaniu danych do zadania, to mamy możliwość wykonania rysunku, narysowania kształtu, wprowadzenia kształtu w oknie aplikacji. Na podstawie tego kształtu możemy potem oteksturować obiekt. W programie można wybierać kolor którym rysujemy, kształt, wypełnienie, a także teksturę która ma być użyta.

3. Wykorzystane komendy:

Podstawową komendą wykorzystywaną nie tylko w tym programie, ale także i w tym jest komenda GLContext która odpowiada za wyrenderowanie kotekstu, w moim kodzie potem jest przypisanie do niej makeCurrent co oznacza ze za context ustaw ten obecny. URL i BufferedImage to dwa typy które powiązane są z biblioteką IO i odpowiadają za pobranie obrazków. Kolejna komenda to texParameteri(), ta komenda odnosi się do nakładania tekstur na obrazek, szczegółów gdzie można użyć gl_texture_wrap_s albo _t odpowiada to za współrzędne tak jak mamy x i y to tutaj s i t. Kolejne polecenie to glEnable, które umożliwia użycie tekstury, a polecenie glTexImage2D umożliwia załadowanie tekstury. Kolejne polecenie to glBindTexture() które odpowiada za przełączanie się jednego obrazu tekstury na inny.
Poniższy kod prezentuje zmiane tekstury obiektu na tę z źródła:

```
GLContext context = displayGL.getContext();
boolean temp = false;
if (!context.isCurrent()) {
    context.makeCurrent();
    temp = true;
}
GL2 gl2 = context.getGL().getGL2();
URL url = this.getClass().getClassLoader().getResource(resourceName);
BufferedImage img = ImageIO.read(Objects.requireNonNull(url));
Texture tex;
tex = AWTTextureIO.newTexture(displayGL.getGLProfile(), img, true);
tex.setTexParameteri(gl2, GL2.GL_TEXTURE_WRAP_S, GL2.GL_CLAMP);
tex.setTexParameteri(gl2, GL2.GL_TEXTURE_WRAP_T, GL2.GL_CLAMP);
if (temp) {
    context.release();
}
return tex;
}
```

Natomiast to odpowiada za załadowanie tekstury z pola rysującego:

```
GLContext context = displayGL.getContext();
boolean needsRelease = false;
if (!context.isCurrent()) {
    context.makeCurrent();
    needsRelease = true;
}
GL2 gl2 = context.getGL().getGL2();
Texture tex;
BufferedImage img = paintPanel.copyOSC();
tex = AWTTextureIO.newTexture(displayGL.getGLProfile(), img, true);
tex.setTexParameteri(gl2, GL2.GL_TEXTURE_WRAP_S, GL2.GL_REPEAT);
tex.setTexParameteri(gl2, GL2.GL_TEXTURE_WRAP_T, GL2.GL_REPEAT);

if (needsRelease) {
    context.release();
}

return tex;
```

A tutaj załadowanie i poprawne przełączanie tekstur

```
Texture textu = currentTexture;
if (textu != null) {
    textu.enable(gl2);
    textu.bind(gl2);
    drawCurrentShape(gl2);
    textu.disable(gl2);
} else
    drawCurrentShape(gl2);
//drawCurrentShape(gl2);
```

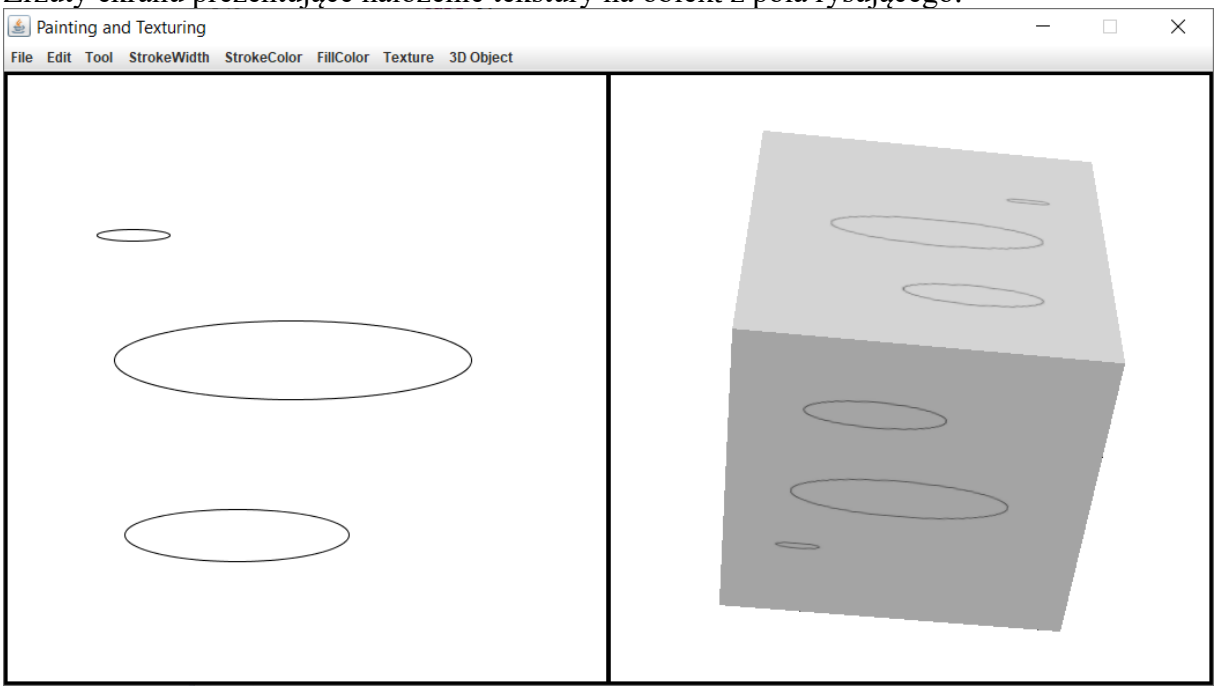
Fragment odpowiadający za wybranie źródła tekstury (obrazka):

```
            break;
        case 3:
            try {
                currentTexture = textureFromResource("earth.jpg");
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            break;
        case 4:
            try {
                currentTexture = textureFromResource("brick.jpg");
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            break;
        case 5:
            try {
                currentTexture = textureFromResource("clouds.jpg");
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
```

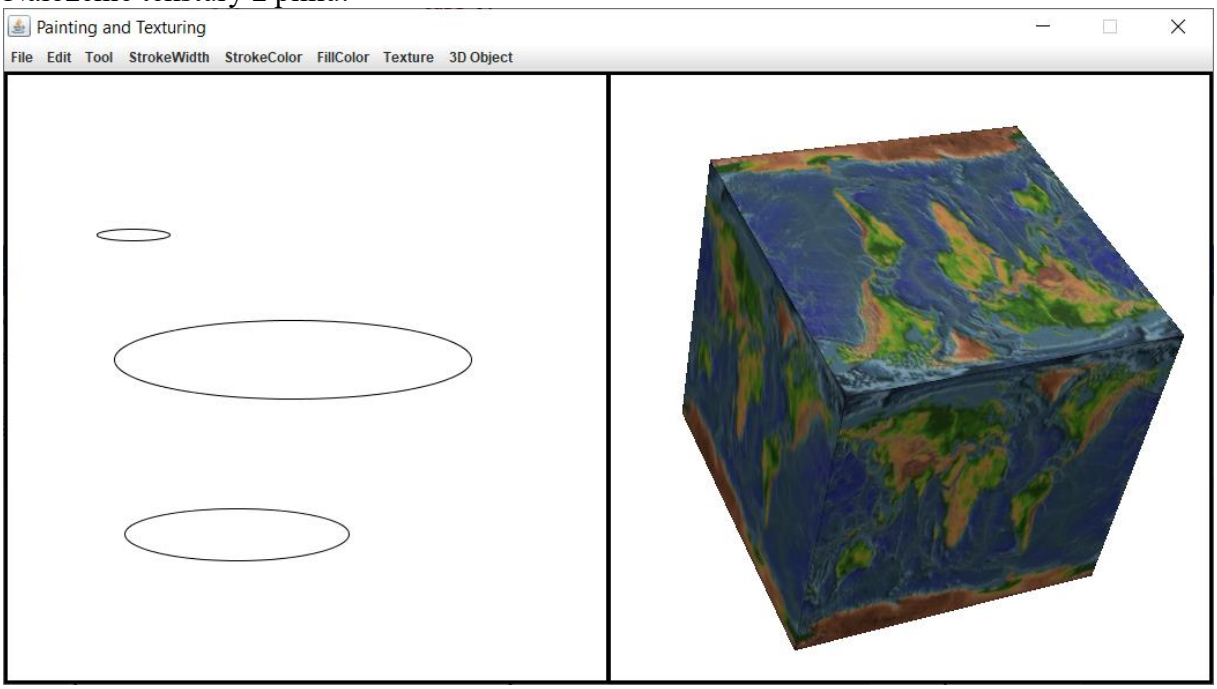
Link do zdalnego repozytorium zawierającego projekt:
https://github.com/Jarverr/8G

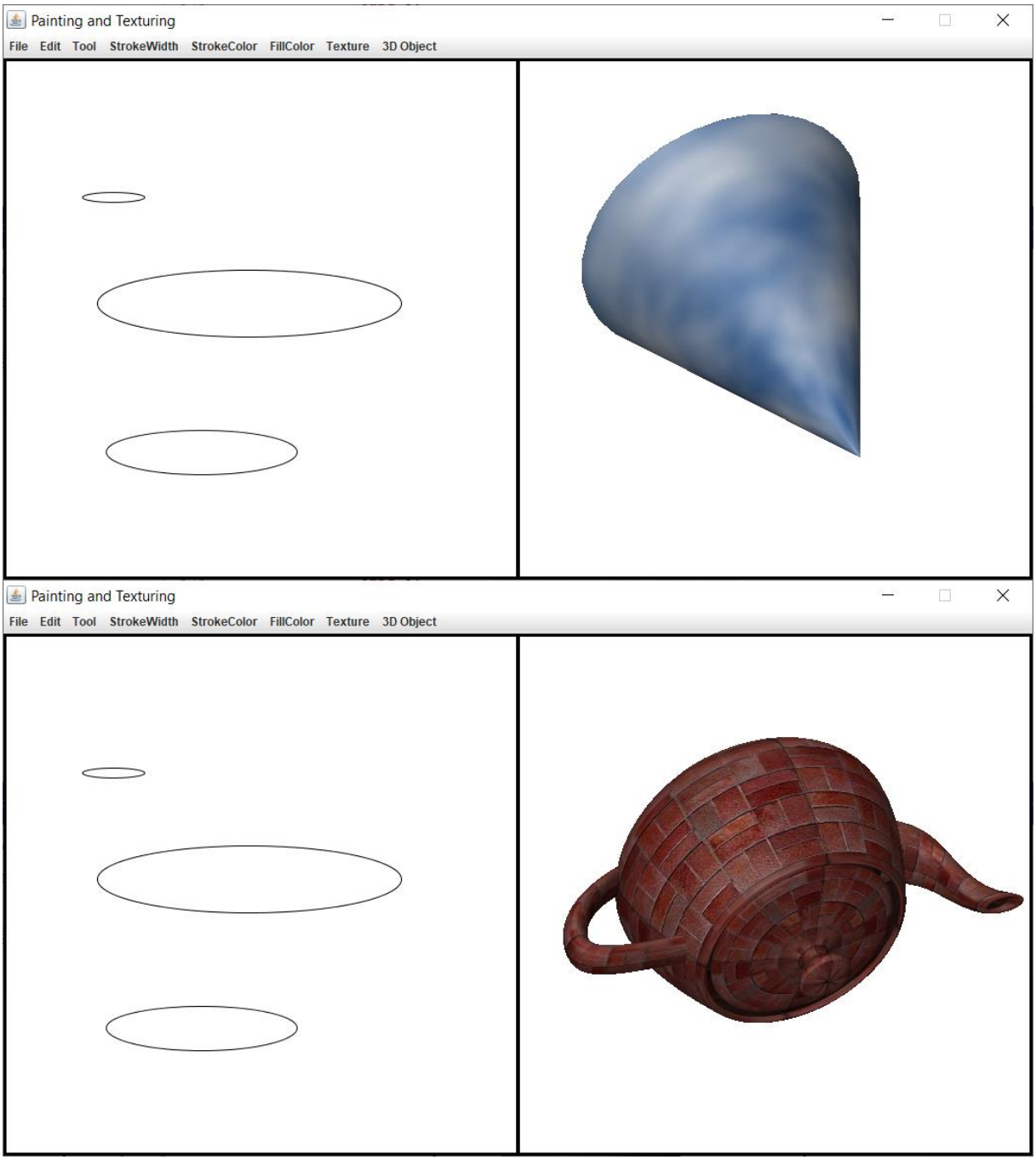
4. Wynik działania:

Zrzuty ekranu prezentujące nałożenie tekstury na obiekt z pola rysującego:

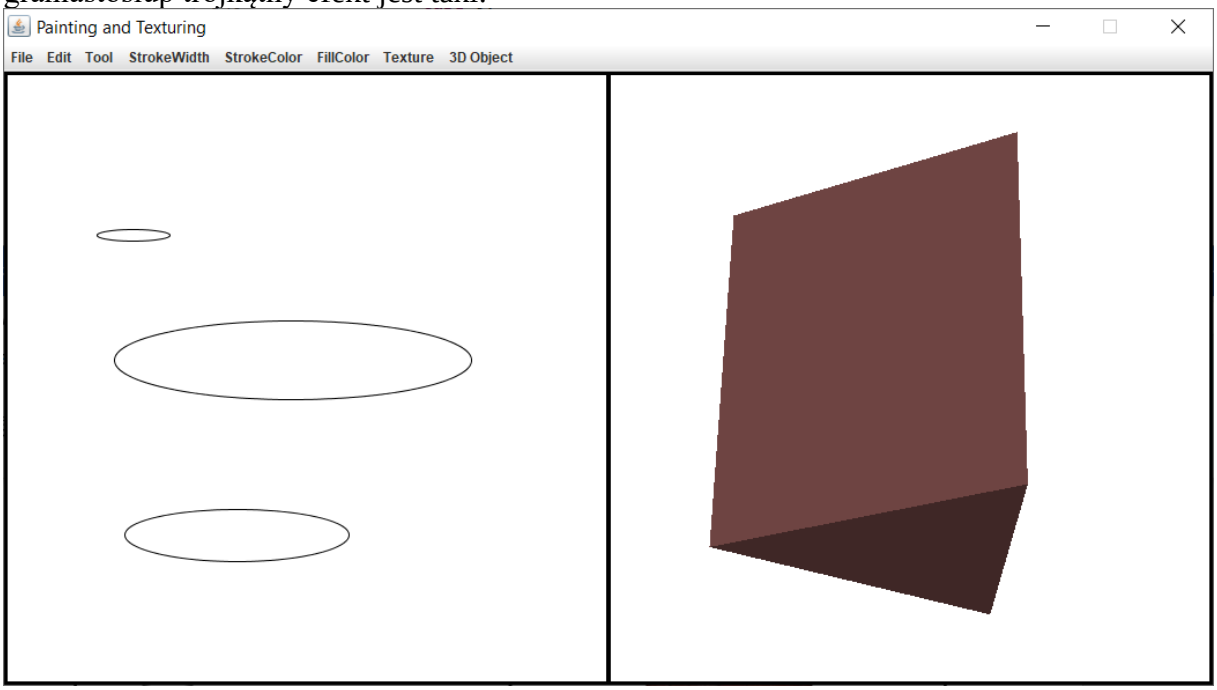


Nałożenie tekstury z pliku:




Painting and Texturing
File Edit Tool StrokeWidth StrokeColor FillColor Texture 3D Object
Painting and Texturing
File Edit Tool StrokeWidth StrokeColor FillColor Texture 3D Object

Nie udało się jedynie przygotować tak programu by dokładnie nakładał teksturę na graniastosłup trójkątny efekt jest taki:



5. Wnioski:

Dzięki możliwością jakie daje biblioteka OpenGL, mogę pobierać tekstury z różnych źródeł a następnie nakładać je na obiekty z różnym skutkiem. Tekstury przygotowane nie muszą być idealne i nie muszą pokrywać całych powierzchni gdyż mogę je np. powtarzać na stworzonym obiekcie, mogę też wpływać na niektóre właściwości danej tekstury. Istnieje też możliwość wycinania konkretnego fragmentu tekstury i umieszczania go na obiekcie, choć niestety to mi się nie udało.