

# **SPRAWOZDANIE**

**Zajęcia: Grafika komputerowa**

**Prowadzący: prof. dr hab. Vasyl Martsenyuk**

**Laboratorium 10**

**28 VI 2021 r.**

**Temat: "Podstawy WebGL/GLSL"**

**Wariant:**

**Liczba kątów - 11**

**Przemysław Garbarczyk**

**Informatyka I stopień**

**Stacjonarne, 4 semestr**

**Gr. 2B**

## 1. Polecenie

Program Program w **lab11.html** pokazuje wiele ruchomych czerwonych kwadratów, które odbijają się od krawędzi płótna. Płótno wypełnia cały obszar zawartości przeglądarki internetowej. Kwadraty odpowiadają również myszy: Jeśli klikniesz lewym przyciskiem myszy lub klikniesz lewym przyciskiem myszy i przeciągniesz na płótnie, cały kwadrat będzie kierowany w stronę pozycji myszy. Jeśli klikniesz lewym przyciskiem myszy, dane punktów zostaną ponownie zainicjowane, więc zaczną się od środka. Możesz wstrzymać i ponownie uruchomić animację, naciskając spację.

Kwadraty są w rzeczywistości częścią jednego prymitywu WebGL typu `gl.POINTS`. Każdy kwadrat odpowiada jednemu z wierzchołków pierwotnego. Oczywiście renderowanie jest wykonywane przez moduł shadera wierzchołków i moduł shadera fragmentu. Kod źródłowy shaderów jest w dwóch fałszywych „skryptach” w górnej części pliku `html`.

Będziesz modyfikował kod modułu shadera i kod JavaScript, aby zaimplementować kilka różnych stylów dla prymitywu punktu. Na przykład możliwe będzie rysowanie kwadratów w różnych kolorach, rysowanie wielokątów zamiast kwadratów i tak dalej. Użytkownik będzie kontrolował program, naciskając klawisze na klawiaturze. Do ciebie należy decyzja, których klawiszy użyć, ale proszę udokumentować interfejs w odpowiednim komentarzu do funkcji `doKey()` lub na górze programu.

Program ma dwie funkcje, nad którymi będziesz musiał pracować: Funkcja `initGL()` jest wywoływana, gdy program jest uruchamiany po raz pierwszy, a funkcje `updateForFrame()` i `render()` są wywoływane dla każdej ramki animacji. Ten sam zestaw poleceń byłby legalny we wszystkich tych poleceniach, ale `initGL()` jest najlepszym miejscem do ustawiania rzeczy, które nie zmieniają się w trakcie działania programu, takich jak położenie zmiennych i zmiennych atrybutów w module shadera; `updateForFrame()` jest przeznaczony do aktualizacji zmiennych JavaScript, które zmieniają się z ramki na ramkę; i `render()` ma na celu wykonanie rzeczywistego rysunku WebGL ramki.

## Atrybut koloru

W oryginalnej wersji programu wszystkie kwadraty są czerwone. Pierwsze ćwiczenie polega na umożliwieniu przypisania innego koloru do każdego kwadratu. Ponieważ kwadraty są naprawdę wierzchołkami w pojedynczym prymitywie typu `gl.POINTS`, można użyć zmiennej atrybutu dla koloru. Atrybut może mieć inną wartość dla każdego wierzchołka. Pierwszym zadaniem jest dodanie zmiennej kolorowej typu `vec3` do modułu shadera wierzchołka i użycie wartości atrybutu do pokolorowania kwadratów. Będziesz także musiał pracować po stronie JavaScript. Będziesz potrzebował `Float32Array` do przechowywania wartości kolorów po stronie JavaScript, a będziesz potrzebował bufora WebGL dla tego atrybutu. Program ma już jeden atrybut, który jest używany do współrzędnych wierzchołków. Będziesz robił coś podobnego do atrybutu `color` (poza tym, że możesz to zrobić w `initGL()`, ponieważ wartości kolorów nie zmieniają się po ich utworzeniu). Można użyć losowych wartości w zakresie od 0,0 do 1,0 dla składników koloru. Po uruchomieniu wielokolorowych kwadratów powinieneś ustawić kolory jako opcjonalne. Możesz włączać i wyłączać użycie tablicy wartości atrybutów za pomocą następujących poleceń, gdzie `a_color_loc` to identyfikator atrybutu `color` w programie shader:

```
gl.enableVertexAttribArray(a_color_loc); // użyj bufora atrybutów
kolorów
gl.disableVertexAttribArray(a_color_loc); // nie używaj bufora
```

Gdy tablica atrybutów jest włączona, każdy wierzchołek otrzymuje swój własny kolor z bufora atrybutów. Gdy tablica atrybutów jest wyłączona, wszystkie wierzchołki otrzymują ten sam kolor, a tę wartość można ustawić za pomocą rodziny funkcji `gl.vertexAttrib *`. Na przykład, aby ustawić wartość używaną, gdy tablica atrybutów kolorów jest wyłączona, można użyć

```
gl.vertexAttrib3f(a_color_loc, 1, 0, 0); // ustaw kolor attribute na
czerwony
```

Pozwól użytkownikowi na naciśnięcie określonego klawisza, aby włączyć lub wyłączyć losowe kolory. Program ma funkcję `doKey()`, która jest już skonfigurowana do reagowania na wprowadzanie z klawiatury. Będziesz

dodawać do programu kilka typów interakcji z klawiaturą. Aby odpowiedzieć na klawiszę, musisz znać numeryczny kod klawiszy. Funkcja `doKey()` wysyła kod do konsoli za każdym razem, gdy użytkownik uderza klawisz, i możesz użyć tej funkcji, aby odkryć wszystkie inne kody klawiszy, których potrzebujesz.

#### Styl punktów

Powinieneś dodać opcję używania stylu wyświetlania dla punktów w postaci wielokąta. Pozwól użytkownikowi wybrać styl za pomocą klawiatury; na przykład, naciskając klawisze numeryczne.

Style będą musiały zostać zaimplementowane w shaderze fragmentu, a będziesz potrzebował nowej zmiennej jednolitej, aby powiedzieć modułowi shadera fragmentu, którego stylu użyć. Dodaj jednolitą zmienną typu `int` do shadera fragmentu, aby kontrolować styl punktu, i dodaj kod do modułu cieniującego fragmentu, aby zaimplementować różne style. Będziesz także musiał dodać zmienną po stronie JavaScript dla lokalizacji zmiennej jednolitej, a będziesz musiał wywołać `glUniform1i`, gdy chcesz zmienić styl.

Naprzykład, żeby narysować punkt jako dysk, odrzucając niektóre piksele:

```
float dist = distance( vec2(0.5), gl_PointCoord );
if (dist > 0.5) {
    discard;
}
```

Powinieneś również wykorzystać przezroczystość alfa w niektórych stylach. Aby umożliwić korzystanie ze składnika `alpha`, musisz dodać następujące linie do funkcji `initGL()`:

```
gl.enable(gl.BLEND);
gl.blendFunc(gl.SRC_ALPHA, gl.ONE_MINUS_SRC_ALPHA);
```

Dzięki tym ustawieniom wartość alfa koloru będzie używana do przezroczystości w zwykły sposób. W szczególności jeden z twoich stylów powinien pokazywać punkt jako wielokąt, który zanika z całkowicie nieprzezroczystego w środku wielokąta do całkowicie przezroczystego na krawędzi.

## 2. Wprowadzam dane:

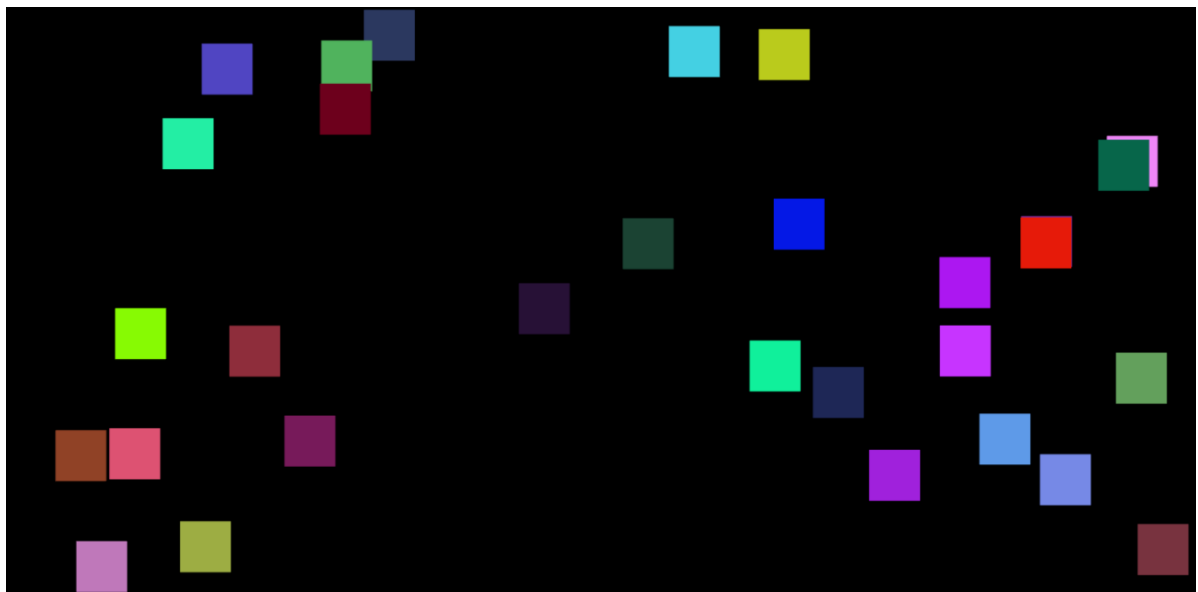
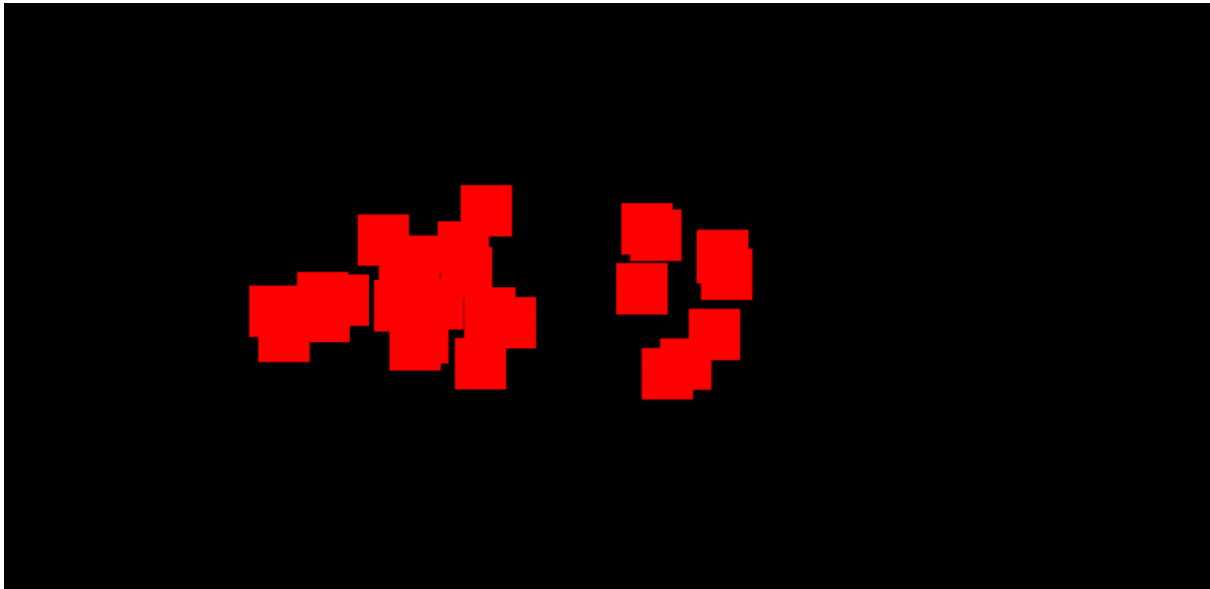
Liczba kątów – 11

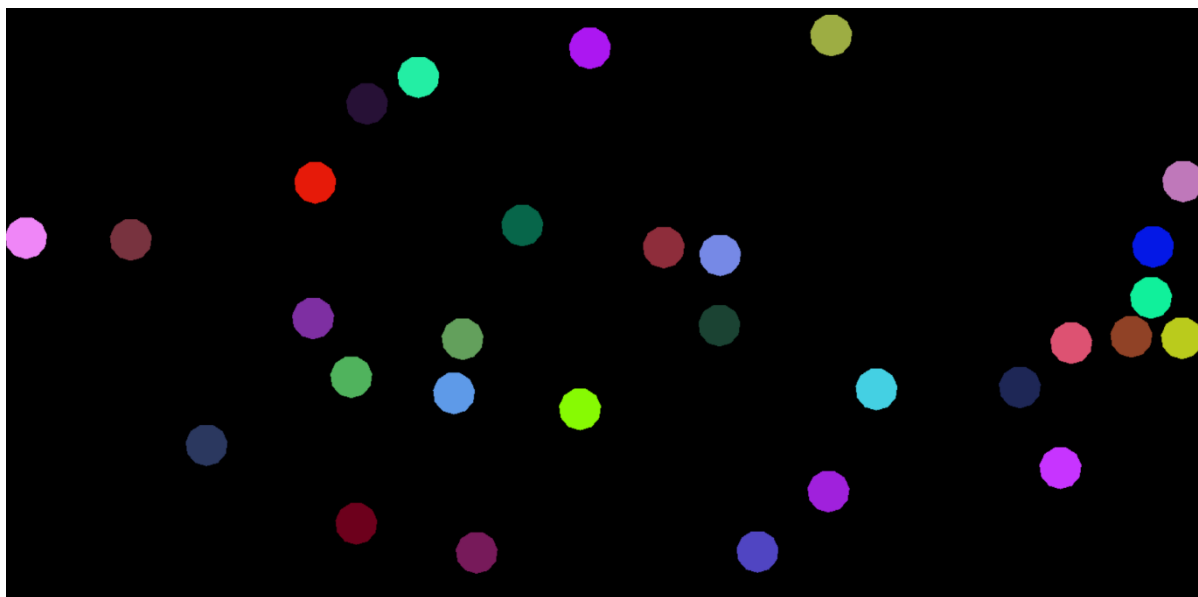
### 3. Wykorzystane komendy:

Kod źródłowy: [LAB GK/Lab10.html at master · Sporemaniak1/LAB GK \(github.com\)](https://github.com/Sporemaniak1/LAB_GK/blob/master/LAB_GK/Lab10.html)

Link do zdalnego repozytorium: [Sporemaniak1/LAB\\_GK \(github.com\)](https://github.com/Sporemaniak1/LAB_GK)

### 4. Wyniki działania:





## 5. Wnioski

Na podstawie otrzymanego wyniku można stwierdzić, że:

- Można ustawić kolory wielokątów;
- Można stwierdzić ilość boków wielokątów.