WSTĘP: - w międzyczasie można wyświetlać przykładowe shadery z shadertoy

Nawet najbardziej szczegółowe modele wyrenderowane przez komputer składają się z prymitywnych kształtów - najczęściej są to trójkąty.

Do manipulacji wyglądem tych modeli - zmiany ich położenia, rotacji, skali, koloru oświetlenia czy jeszcze innych efektów graficznych służą shadery.

Program cieniujący - shader to program komputerowy, który wykonuje się na karcie graficznej.

Programy te działają na różnych poziomach, w zależności od rodzaju shadera są to wierzchołki, lub piksele.

Obliczenia wykonują się równolegle dla wszystkich wierzchołków/pikseli.

Karty są przystosowanie do przeprowadzania wielu obliczeń równolegle - w tym samym czasie

dzięki czemu możemy renderować grafikę w czasie rzeczywistym - procesor nie jest do tego przystosowany.

Proces przetwarzania danych o wierzchołkach jest dość skomplikowany, my skupimy się na najważniejszych, czyli na etapie, w którym wykorzystywane są vertex shader i fragment shader.

Vertex shadery za przekształcenie wierzchołków, ich położenia, skalowanie, rotację. W efekcie, na wyjściu shadera znajdują się przekształcone współrzędne wierzchołków na podstawie których można stwierdzić, które wierzchołki (czyli modele, na które się składają) będą widocznie na ekranie. Dane o tych wierzchołkach są przekazywane do fragment shadera.

Na dzisiejszych laboratoriach zajmiemy się właśnie fragment shaderem

Fragment shader działa na pikselach, które są generowane przez proces renderowania obiektów na scenie. Jego głównym zadaniem jest określanie ostatecznego koloru (oraz innych właściwości) każdego piksela na podstawie różnych czynników, takich jak oświetlenie, tekstury, cienie, przezroczystość i wiele innych.

Fragment shader otrzymuje dane wejściowe takie jak pozycję piksela, kolory tekstur, dane oświetleniowe, a także dodatkowe wartości przekazywane z vertex shadera. Na podstawie tych danych programista może implementować różne operacje, takie jak miksowanie kolorów, generowanie efektów specjalnych, symulowanie oświetlenia i wiele innych.

Vertex oraz fragment shader są tylko częścią procesu zwanego potokiem renderowania, np. między vertex a fragment shaderem jest jeszcze etap rasteryzacji, czyli podzielenia tego co zostanie wyświetlone na piksele.

SHADERTOY:

Na komputerach powinniście mieć włączoną przeglądarkę wraz ze stroną shadertoy, jeśli nie to wpiszcie tę frazę w google i wejdźcie pod pierwszy link.

Jest to platforma online służąca do tworzenia własnych fragment shaderów. Opiera się o WebGL - webową wersję openGL'a pozwalającą na renderowanie grafiki w przeglądarce internetowej (openGL - standard programowania w grafice, nie bilbioteka!)

// TEN FRAGMENT TYLKO DO PODSTAWÓWKI - chyba, że będziemy robić ze szkołami średnimi zadania dla podstawówek

// FRAGMENT DLA ŚREDNICH TROCHĘ NIŻEJ

W naszych przykładowych zadaniach będziemy korzystać z wcześniej przygotowanych przez nas funkcji pozwalających rysować prymitywne kształty (koło, prostokąt, trójkąt), zatem nie martwcie się, zadania będą bardzo proste i cały czas jesteśmy gotowi do pomocy.

Jednak zanim zaczniemy, pokażemy wam podstawy, które pozwolą wam wykonać zadania:

Po pierwsze, w grafice komputerowej bardzo często korzysta się z wektorów, więc nic niespodziewanego, że język GLSL umożliwia obsługę wektorów:

Dzieciaki prawdopodobnie nie będą wiedziały co to wektor – trzeba wytłumaczyć

wytłumacz co to wektor ale tak, żeby zrozumiał 10 latek

Oczywiście! Wektor to taki specjalny "strzałka" w przestrzeni, która ma długość (jakbyś ją mierzył linijką) i kierunek (w którą stronę pokazuje). Możesz sobie wyobrazić wektor jako strzałkę na mapie, która pokazuje, jak dotrzeć z jednego miejsca do drugiego.

Pomyśl o tym tak: Wyobraź sobie, że jesteś w punkcie A na mapie i chcesz dotrzeć do punktu B. Teraz wskazując palcem w kierunku punktu B, możesz sobie wyobrazić, że ta linia, którą wyznacza twój palec, to właśnie wektor. Linia ta ma długość, która pokazuje jak daleko musisz się przemieścić, aby dotrzeć do punktu B.

Ale wektor może pokazać również kierunek w przestrzeni trójwymiarowej. Na przykład, gdy patrzysz na drzewo, możesz sobie wyobrazić wektor wskazujący na górę, w kierunku gałęzi. Ten wektor będzie miał długość, która pokazuje, jak daleko musisz się przesunąć, aby dotrzeć do górnych części drzewa.

Wektory są bardzo przydatne w matematyce, fizyce i innych dziedzinach nauki. Pomagają nam rozumieć i opisywać ruch, siły, prędkości, a także wiele innych zjawisk w świecie wokół nas.

Najważniejsze żeby zrozumieli, że wektor ma współrzędne, które w przypadku grafiki komputerowej reprezentują składowe kolorów RGBA.

to można na rzutniku wyświetlać, np. w edytorze shadertoy

w celu stworzenia takiego wektora musimy napisać np:

vec2 fajny_wektor = vec2(0.0,0.0) - w ten sposób tworzymy wektor 2 wymiarowy o nazwie fajny_wektor o współrzędnych x = 0.0 y = 0.0

ale można też inaczej:

vec2 fajny_wektor2 = vec2(0.0) - taki skrócony zapis oznacza, że wszystkie współrzędne tego wektora będą równe wartości podanej w nawiasie

możemy się dostawać do poszczególnych składowych wektora poprzez np:

float x = fajny_wektor.x; - do zmiennej x przypisujemy wartość współrzędnej x wektora

Oczywiście możemy tworzyć też wektory 3 i 4 wymiarowe np:

vec3 kolor = vec3(0.0,1.0,0.0) - tworzymy wektor 3 wymiarowy o nazwie kolor

Wyjaśnienie: kolory w grafice komputerowej składają się z 4 kanałów - RGBA, wartość każdego z nich jest reprezentowana przez liczbę z przedziału (0.0,1.0)

Aby poprawie ustawić kolor we fragment shaderze potrzebujemy więc wektora 4 wymiarowego. Każdy fragment shader ma za zadanie przypisać taką wartość do zmiennej środowiskowej

fragColor, która jest wektorem 4 wymiarowym właśnie. Mając do dyspozycji wektor 3 wymiarowy (wyżej zdefiniowany kolor), możemy postąpić w następujący sposób:

fragColor = vec4(kolor,1.0);

taki zapis ustawia pierwsze 3 współrzędne (xyz, bądź jak kto woli rgb) wektora kolor, a jako ostatnią współrzędną (alpha) ustawia 1.0 - czyli brak przezroczystości.

Jest to ogólna zasada konstrukcji wektorów, zatem takie zapisy również będą poprawne:

vec2 a = vec2(1.0,0.0);

vec4 b = vec4(0.0,a,0.0) - w tym przypadku współrzędne wektora a zostaną przypisane do współrzędnych y i z wektora b

vec4 c = vec4(a.y,a.x,0.0,0.0) - w tym przypadku współrzędna x wektora c będzie równa współrzędnej y wektora a itd...

W jednym z zadań będzie konieczne sprawdzenie długości wektora, służy do tego funkcja length(wektor), gdzie wektor może być 2,3 lub nawet 4 wymiarowy.

- siatka na shadertoy:

Po uruchomieniu przykładowych programów wyświetli się wam pewien obraz, a za nim siatka, ta siatka reprezentuje współrzędne na ekranie, gdzie każda kratka reprezentuje skok co 0,1

Początek tego układu znajduje się w lewym dolnym rogu - punkt (0,0), natomiast w prawym górnym mamy punkt (1,1). Nie przejmujcie się, że współrzędne na osi X i Y mają różne odstępy między sobą, wynika to z tego, iż wyświetlacz nie jest kwadratowy, tylko prostokątny, a co za tym idzie układ współrzędnych musiał zostać przeskalowany.

FRAGMENT DLA SZKÓŁ ŚREDNICH:

Można powiedzieć, że fragment shader składa się z funkcji mainImage oraz funkcji zdefiniowanych przez użytkownika. Funkcja mainImage musi ustawić fragColor (vec4) na kolor przetwarzanego piksela, a funkcje, które będą nam potrzebne to Mandelbrot(vec2 liczba) oraz kwadrat(vec2 liczba). Pokazać jak tworzy się wektory - patrz fragment dla podstawówki. Można wspomnieć o pozostałych rzeczach dla podstawówki (zakres kolorów, funkcja length), ale nie są one potrzebne do wykonania zadania.

Liczby zespolone - prezentacja

Fraktale:

Fraktal (łac. fractus – złamany, cząstkowy, ułamkowy) w znaczeniu potocznym oznacza zwykle obiekt samopodobny (tzn. taki, którego części są podobne do całości)[1] albo „nieskończenie złożony” (ukazujący coraz bardziej złożone detale w dowolnie wielkim powiększeniu).

PRZEPROWADZENIE ZADAŃ:

na samym początku pokazać, gdzie się klika kompiluj na stronie, powiedzieć co zrobić w przypadku błędu (nwm podnieść rękę czy coś)

zad 1)

przekopiować zawartość pliku _podst1.glsl / na gicie PODSTAWOWKA/funkcje bez obrotow/

jeszcze raz zapoznać z działaniem kolorów, zakres, zapis wektorowy

pokazać, gdzie zmienić kod i jakiego efektu mają się spodziewać - zmiana koloru koła

pokazać jak zmienić kolor na kolor odpowiadający pozycji myszki – w drugim docx

można zadać pytanie, dlaczego akurat takie kolory się wyświetlają po kliknięciu w danym miejscu (np. w lewym dolnym rogu (punkt 0,0) kolor czarny bo takie są współrzędne, lewy górny czerwony bo (1,0) itd.

zad 2.1)

przekopiować zawartość pliku _podst2 wariant 1.glsl / na gicie PODSTAWOWKA/funkcje bez obrotow/

przypomnieć jak działa wyświetlany układ współrzędnych (0,0) - lewy dolny, (1,1) -prawy górny, skok co 0.1.

pokazać, gdzie mają zmieniać wartośći (podkreślić, że współrzędne z zakresu <0,1>).

Jeśli ktoś by spytał, dlaczego promień koła jest inny niż wpisał to wytłumaczyć, że to przez skalowanie osi X:

chodzi o to, że wyświetlacz jest prostokątny, więc jednostka na X jest inna niż na Y, promień koła jest wyznaczany względem osi Y (nieprzeskalowanej), zatem jeśli wpisano np. radius = 0.1; to promień wyniesie 1 jednostkę ale osi Y. Jeśli ktoś bardzo chciałby, żeby promień był liczony względem osi X to wystarczy przemnożyć promień przez zmienną ratio: radius = 0.1 * ratio; gdzie ratio to stosunek rozdzielczości Y do X.

Na koniec pokazać, który kod odkomentować aby rozpocząć animację samochodu - jeśli obraz nadal będzie statyczny to trzeba kliknąć play w lewym dolnym wyświetlacza na stronie

zad 2.2)

przekopiować zawartość pliku _podst2 wariant 2.glsl / na gicie PODSTAWOWKA/funkcje bez obrotow/

Wytłumaczyć jak działa sprawdzenie, czy punkt należy do okręgu o danym środku i promieniu - rysunek w docx

Jeśli dzieciaki nie miały wcześniej styczności z programowaniem, wytłumaczyć jak działa konstrukcja if oraz operatory porównania > < >= <=

pokazać fragment funkcji inCircle, który jest do zmiany

ponownie można pokazać, jak włączyć animację

zad liceum):

przekopiować zawartość pliku mandelbrot.glsl / na gicie w folderze LICEUM

wszystkie najważniejsze informacje na te tematy są w docx na gicie:

krótko czym są liczby zespolone

jak się dodaje, odejmuje mnoży

czym jest zbiór mandelbrota - najważniejsze żeby zrozumieli wzór rekurencyjny, bo na jego uzupełnieniu polega zadanie

pokazać, gdzie w kodzie znajduje się miejsce do uzupełnienia

powiedzieć, że liczba zespolona jest w naszym przypadku reprezentowana przez wektor 2 wymiarowy (część rzeczywista, część urojona)

funkcja kwadrat(liczba) podnosi liczbę zespoloną do kwadratu i zwraca ją.

jeśli ktoś będzie ciekawy jak działa funkcja Mandelbrot(vec2 liczba):

nasz wyświetlacz traktujemy jako płaszczyznę zespoloną - w tym przypadku jednak punkt (0,0) znajduje się po środku a nie w lewym dolnym rogu. Każdy z punktów na tej płaszczyźnie trafia do funkcji Mandelbrot, następnie w pętli for testujemy maxIterations (200) razy, czy wzór rekurencyjny nie ucieka do nieskończoności (inaczej, czy punkt należy do zbioru). Następnie jako wynik funkcji przekazywana jest liczba z zakresu 0.0 do 1.0 w zależności od tego, ile iteracji pętli for się wykonało. Punkty wewnątrz fraktala to punkty, dla których pętla for wykonała się maksymalną liczbę razy - należą do zbioru mandlebrota

W pewnym momencie - przy zbyt dużym przybliżeniu na ekranie pojawi się "pikseloza", jest to spowodowane ograniczoną dokładnością reprezentacji liczb zmiennoprzecinkowych w komputerze.

Da się to ominąć, ale wykracza to daleko poza zakres tego laboratorium.