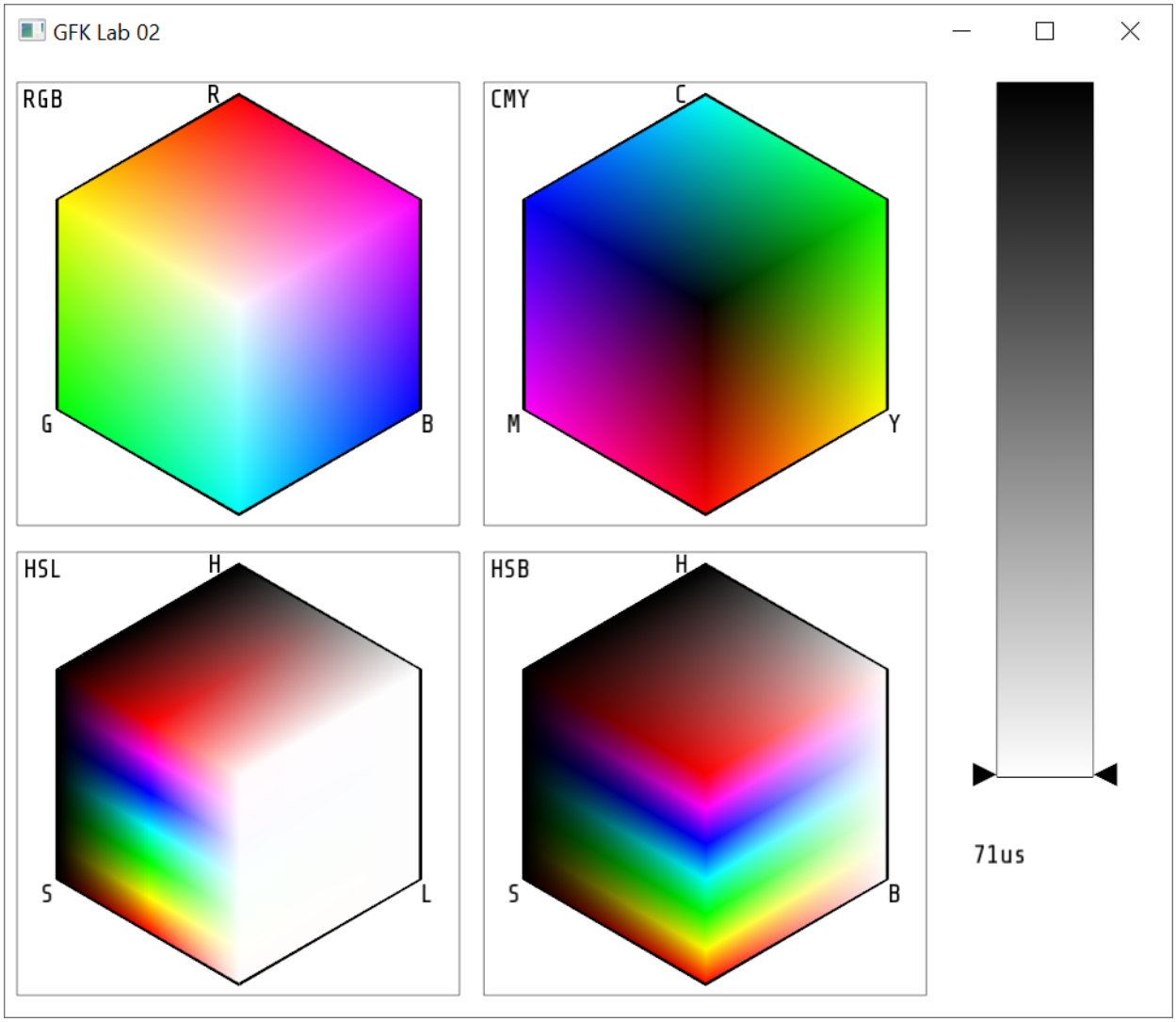
Modele i przestrzenie barw.

Zadanie

Napisać aplikację, która na ekranie wyrysuje cztery sześciokąty barw oraz „suwak” pozwalający ustawić na przykład parametr L w sześciokącie HSL. Każdy sześciokąt będzie przedstawiał barwy w innej reprezentacji (innym modelu) zgodnie z poniższym rysunkiem:



Przy rozwiązaniu zadania należy zwrócić szczególną uwagę na szybkość wykonywania się programu. Można to łatwo sprawdzić mierząc czas potrzebny na rysowanie. Proszę do pomiaru wydajności wykorzystywać metodę z załączonego przykładu aby można było łatwo porównać wyniki. Czas podawany jest w mikrosekundach. Proszę zwrócić uwagę na to aby okienko prawidłowo się skalowało.

Cel

Zapoznanie się z metodami wyliczania barwy w najprostszych modelach barw: RGB, CMY, HSL i HSB. Praktyczne zrozumienie pojęć nasycenia (saturation), koloru (hue), jasności (lightness) i jaskrawości (brightness).

Środki

Dowolne środowisko programistyczne, biblioteka SFML.

Zarys możliwego rozwiązania

Przygotować klasę bazową rysującą sześciokąt w zadanym obszarze. Klasa powinna posiadać konstruktor lub metodę, która ustala lewy górny i prawy dolny narożnik obszaru, w którym sześciokąt ma zostać narysowany. Metoda ta powinna ustalić wszystkie parametry rysowanego obrazu. Klasa powinna zawierać metodę rysującą ramkę sześciokąta oraz wirtualną metodę wypełniającą wnętrze sześciokąta. Następnie z klasy bazowej należy wyprowadzić klasy potomne, w których metoda wirtualna zostanie zastąpiona metodami konkretnymi wyliczającymi składowe RGB na podstawie modelu RGB, CMY, HSL i HSB dla każdego punktu sześciokąta.

Przygotowany fragment programu

Przygotowane źródła zawierają klasę **`hexagon`**, która realizuje już sporą część zadania. Punkty **`p[0]`**,…, **`p[5]`** to wyliczone już dla Państwa wierzchołki sześciokąta. **`p[0]`** to punkt położony u góry zaś kolejne liczone są z kierunkiem przeciwnym do ruchu wskazówek zegara. Oprócz tego w kodzie umieszczono funkcję **`rhombus(...)`**, która zwraca informację o położeniu punktu względem rombu zgodnie z rysunkiem poniżej. Funkcję tą można wykorzystać do rysowania punktów wewnątrz sześciokąta. Sześciokąt można zbudować z trzech odpowiednio ustawionych rombów. Definicja funkcji wygląda następująco:

```
rhombus(sf::Vector2f p0, sf::Vector2f p1, sf::Vector2f p,
        float& alpha, float& beta)
```

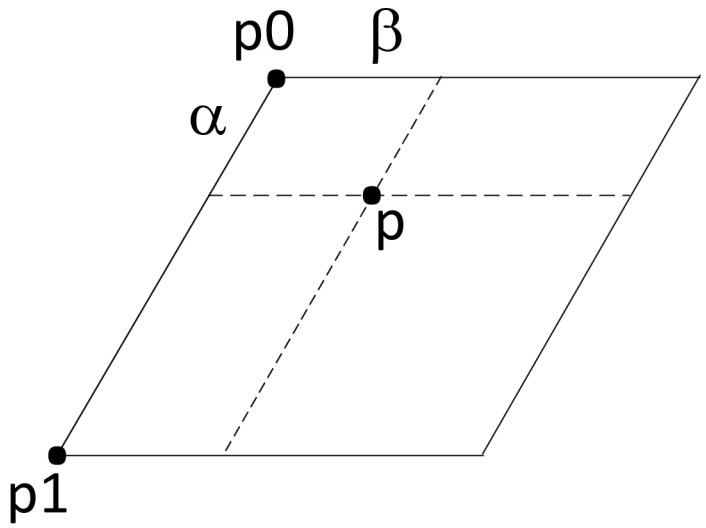
gdzie zmienne:

`p0,p1` - opisują położenie dwóch kolejnych wierzchołków rombu

`p` - to współrzęda punktu wewnątrz rombu

Do funkcji przekazywane są poprzez referencję dwie zmienne: **`alpha`** i **`beta`**. Funkcja ustala proporcje w jakich rzuty równoległe punktu **`p`** przetną boki rombu i przypisuje je do zmiennych **`alpha`** i **`beta`**.

Funkcja zwraca wartość **`true`** jeżeli punkt **`p`** znajduje się wewnątrz rombu oraz wartość **`false`** w przeciwnym przypadku. Oczywiście źródło programu można absolutnie dowolnie modyfikować w razie potrzeby lub w ogóle z niego nie korzystać. Proszę zwrócić uwagę na komentarze zawarte w kodzie: program w oryginalnej postaci wymaga co najmniej standardu C++17.



Jak się przygotować przed zajęciami

Przed zajęciami proszę zapoznać się z następującymi zagadnieniami: wykonywanie podstawowych operacji graficznych w bibliotece SFML (rysowanie punktu, odczytywanie koloru punktu, rysowanie linii, kopiowanie wycinka obrazu etc). Należy zapoznać się również z metodami obsługi zdarzeń w tej bibliotece. Sposób rysowania kolorowego rombu przedstawiony był na wykładzie pierwszym!

Zadanie tylko dla chętnych (trudne)

Proszę przygotować okrojoną wersję laboratorium zawierającą tylko dwa sześciokąty: RGB oraz CMY. Brak suwaka i możliwości zmiany rozmiaru okna – ma ono ustalone rozmiary: powiedzmy 1024x768. W czasie trwania programu „suwak” (którego oczywiście w tej wersji nie widać) przemieszcza się cyklicznie z góry do dołu i z powrotem. Poniżej wyświetla się imię i nazwisko autora programu (widnieje ono również w belce tytułowej). Cała zabawa polega na tym aby plik wykonywalny był jak najmniejszy. Programy większe niż 4kB (4096 bajtów) nie biorą udziału w zabawie. Programy nie zawierające imienia i nazwiska autora wewnątrz okna – również nie będą brane pod uwagę. Czas na rozwiązanie – do końca trwania laboratoriów (czyli pewnie jakiś maj...).