Programando o Quick LEDs

A WPILib possui uma API para controlar LEDs WS2812 e WS2811 com seu pino de dados conectado via PWM.

Instanciando um objeto Adressable LED

Primeiramente você deve criar um objeto `Addressable LED` que tem como argumento a porta PWM que está conectado. Deve ser uma porta PWM no roboRIO. Em seguida, você define o número de LEDs (pixels) localizados em sua faixa de LEDs, o que pode ser feito com a função setLength().

> É importante saber que definir o comprimento da fita LED é uma tarefa cara e não é recomendado executá-la periodicamente.

Depois que o comprimento da fita for definido, você terá que criar um objeto `AddressableLEDBuffer` que recebe o número de LEDs como entrada. Em seguida, você chamará `myAddressableLed.setData(myAddressableLEDBuffer)` para definir os dados de saída do led. Então, você pode chamar `myAddressableLed.start()` para gravar a saída continuamente. Abaixo está um exemplo completo do processo de inicialização.

```
@Override
public void robotInit() {
  // porta PWM 9
  // deve ser um cabeçalho PWM , não MXP ou DIO
  m_led = new AddressableLED(9);

  // Reutiliza o buffer
  // Padrão para uma fita de 60 pixeis
  // Definir o comprimento da fita é uma função cara no código, então faça apenas uma vez
  m_ledBuffer = new AddressableLEDBuffer(60);
  m_led.setLength(m_ledBuffer.getLength());

  // Define os dados
  m_led.setData(m_ledBuffer);
  m_led.start();
}
```

> O roboRIO suporta apenas 1 objeto AddressableLED. Como os LEDs WS2812B sãoconectados em série, você pode acionar várias faixas conectadas em série a partir do objeto AddressableLED.

Configurando toda a fita para uma cor

A cor pode ser definida para um LED individual da fita usando dois métodos. `setRGB()` que aceita valores RGB como entrada e `setHSV()` que aceita valores HSV como entrada.

Utilizando valores RGB

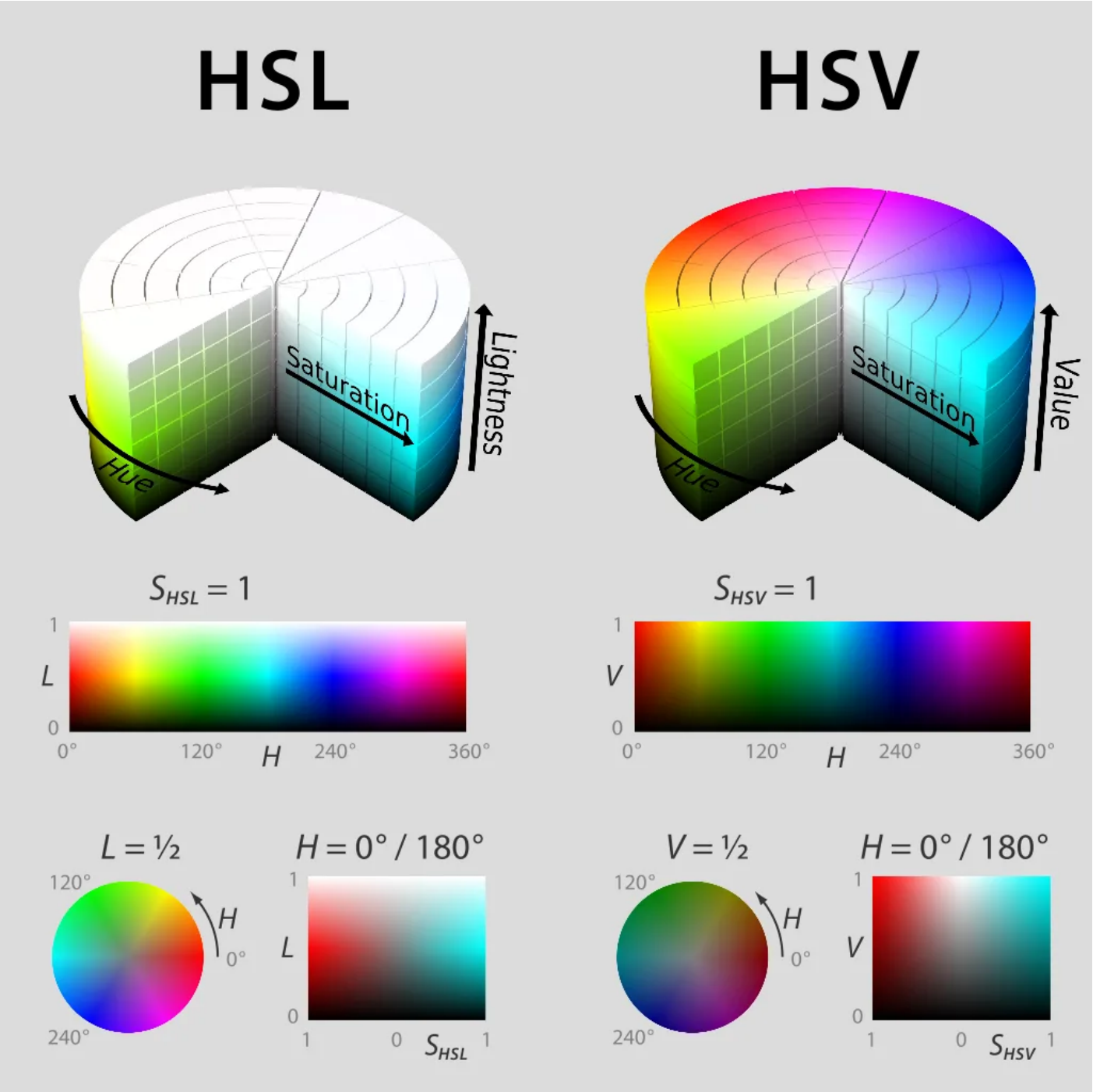
RGB significa Vermelho, Verde e Azul. Este é um modelo de cores bastante comum, pois é bastante fácil de entender. Os LEDs podem ser configurados com o método `setRGB` que leva 4 argumentos: índice do LED, quantidade de vermelho, quantidade de verde, quantidade de azul. A quantidade de Vermelho, Verde e Azul são valores inteiros entre 0-255.

```
 for (var i = 0; i < m_ledBuffer.getLength(); i++) {
// Define os LEDs específicos para a cor vermelha em RGB
  m_ledBuffer.setRGB(i, 255, 0, 0);
 }


 m_led.setData(m_ledBuffer);
```

Utilizando valores HSV

HSV significa Matiz, Saturação e Valor. Matiz descreve a cor ou matiz, sendo a saturação a quantidade de cinza e o valor o brilho. Na WPILib, Matiz é um número inteiro de 0 a 180. Saturação e Valor são números inteiros de 0 a 255. Se você olhar para um seletor de cores como o do Google, Matiz será de 0 a 360 e Saturação e Valor serão de 0% a 100%. É da mesma forma que o OpenCV lida com cores HSV. Certifique-se de que os valores HSV inseridos no WPILib estejam corretos, ou a cor produzida pode não ser a mesma esperada.



Os LEDs podem ser configurados com o método `setHSV` que leva 4 argumentos: índice do LED, matiz, saturação e valor. Um exemplo é mostrado abaixo para definir a cor de uma faixa de LED para vermelho (matiz 0).

```
for (var i = 0; i < m_ledBuffer.getLength(); i++) {
 // Define os LEDs específicos para a cor vermelha em HSV
 m_ledBuffer.setHSV(i, 0, 100, 100);
}

m_led.setData(m_ledBuffer);
```

Criando um efeito arco-íris

O método abaixo faz algumas coisas importantes. Dentro do loop for, ele distribui igualmente o matiz por todo o comprimento do fio e armazena o matiz individual do LED em uma variável chamada matiz. Em seguida, o loop for define o valor HSV desse pixel especificado usando o valor de matiz.

Movendo-se para fora do loop for, o `m_rainbowFirstPixelHue` itera o pixel que contém o matiz “inicial”, criando o efeito arco-íris. `m_rainbowFirstPixelHue` então verifica se o matiz está dentro dos limites de matiz de 180. Isso ocorre porque o matiz HSV é um valor de 0 a 180.

```
private void rainbow() {
  // Para cada pixel
  for (var i = 0; i < m_ledBuffer.getLength(); i++) {
    // Calcule o hue - hue é melhor para arco iris por conta do
    // formato de cor ser um circulo então apenas uma variável é mudada
    // para processar
      final var hue = (m_rainbowFirstPixelHue + (i * 180 / m_ledBuffer.getLength())) % 180;
    // Define o valor
    m_ledBuffer.setHSV(i, hue, 255, 128);
  }
  // Aumenta o hue para fazer o arco iris "se mover"
  m_rainbowFirstPixelHue += 3;
  // Check bounds
  m_rainbowFirstPixelHue %= 180;
}
```

Revisão #3
Criado 30 janeiro 2024 10:27:07 por Luca Carvalho
Atualizado 24 junho 2024 19:41:13 por Bruno Toso