Image Type: HSV

Für viele Anwendungen ist der HSV-Farbraum gegenüber dem RGB-Farbraum vorteilhaft, z.B. wenn bestimmte Farbbereiche in Farbbildern detektiert werden sollen, z.B. Signalfarben von Verkehrsschildern oder Hautfarbe zur Personendetektion. Im HSV-Farbraum kann dann der V-Kanal ignoriert werden, d.h. Schattenwurf oder inhomogene Beleuchtung hat keinen Einfluss auf die Farbe in den Kanälen H und S.
Erweitern Sie Ihre Image-Klasse um den Typ HSV. (enum, Konstructor)

```
1  public: enum TYPE{UNDEF=0,BINARY=1,GRAY=2,RGB=3,HSV=4};
2  private: int SIZE[5] = {0,1,1,3,3}; // bytes per pixel
3  ...
4  Image(int w,int h, TYPE type);
5  void Init(int w, int h, TYPE t);
6  int DataSize()const { return SIZE[type]*width*height; }
```

Das Laden und Speichern von HSV-Bildern macht i.d.R. wenig Sinn, da meistens keine externen Programme zum Betrachten oder Verarbeiten von HSV-Bildern zur Verfügung stehen. Daher müssen Sie Ihre Load- oder Save-Funktion auch nicht erweitern; lediglich beim Versuch, HSV-Bilder zu speichern, ist eine Ausgabe sinnvoll, dass dies nicht möglich ist!

```
1  void Image::Save(){
2    clog << "Image::Save()" << endl;
3    switch(type)
4    {
5      case BINARY: Save(fileName, "P4", width, height, 1, Data); break;
6      case GRAY: Save(fileName, "P5", width, height, 255, Data); break;
7      case RGB: Save(fileName, "P6", width, height, 255, Data); break;
8      case HSV:cerr<<"ERROR! Unable to save image of type HSV!"<<endl;break;
9      case UNDEF:cerr<<"ERROR! Unable to save image of type UNDEF!"<<endl;break
10   }
11 }
```

Einfaches HSV-Testbild

In der nächsten Aufgabe soll die Transformation HSV → RGB implementiert werden. Zum Testen dieser Funktion können Sie sich ein einfaches Testbild wie folgt erstellen:

```
1  clog << "07a create an hsv image with (h,s,v)=(x,y,255)"
2       << "and transform it to rgb" << endl;
3  Image hsv(256,256,Image::HSV);      // HSV test image
4  Image rgb;                          // final result
5  int i = 0;                          // index for 1-dim data access
6  for(int y=0;y<256;++y) for(int x=0;x<256;++x) // all pixels
7  {
8    hsv.Data[i++] = x;                // hue-channel: 0...255
9    hsv.Data[i++] = y;                // saturation-channel: 0...255
10   hsv.Data[i++] = 255;              // value-channel: const/max
11 }
12 hsv2rgb(hsv,rgb);                   // transform test image to rgb
13 rgb.SaveAs("images/rgb");           // ... and save it
```

RGB ↔ HSV

Implementieren Sie folgende Funktionen:

```
1  const unsigned char HUE_UNDEFINED = 255; // hue is in the range of [0,255],
2                                           // where 255 is undefined
3
4  /** Transform from RGB to HSV */
5  void rgb2hsv(const Image& rgb, Image& dst);
6
7  /** Transform from HSV to RGB */
8  void hsv2rgb(const Image& hsv, Image& dst);
9
10 /** Split HSV image into separate channels */
11 void splitHSV(const Image& hsv, Image& h, Image& s, Image &v);
12
13 /** Combine separate HSV channels to an HSV image*/
14 void combineHSV(Image& hsv, const Image& h, const Image& s, const Image &v);
```

Zum Testen dieser Funktionen können sie z.B. wie folgt vorgehen:

```
1  clog << "07c transform an rgb image into hsv, split it into its channels"
2       << "and transform it back" << endl;
3  Image rgb("images/paprika.ppm");
4  Image hsv,h,s,v;
5  rgb2hsv(rgb,hsv);
6  splitHSV(hsv,h,s,v);
7  h.SaveAs("images/hue");
8  s.SaveAs("images/sat");
9  v.SaveAs("images/val");
10 hsv2rgb(hsv,rgb);
11 rgb.SaveAs("images/paprika_copy.ppm");
```