Compte rendu TP image 3D
Traitement de l’image

Tristan Cossin
Master IMAGINA

MAI 2017

Exercice 1 à 3 :

Lecture d’un fichier img et stockage:

```
cout << "Veuillez donner la taille en X de l'image SVP" << endl;
cin >> tailleX;

cout << "Veuillez donner la taille en Y de l'image SVP" << endl;
cin >> tailleY;

cout << "Veuillez donner la taille en Z de l'image SVP" << endl;
cin >> tailleZ;


int taille = tailleX * tailleY * tailleZ;

unsigned short *buffer = new unsigned short[taille];

//on ouvre le fichier
//FILE *image = fopen("images/orange.256x256x64.0.3906x0.3906x1.0.img", "rb");
//FILE *image = fopen("images/INCISIX.512x512x166.0.3613281x0.3613281x0.5.img", "rb");
//FILE *image = fopen("images/t1-head.256x256x129.1.5x1.5x1.5.img", "rb");
FILE *image = fopen("images/whatisit.208x208x123.1.0x1.0x1.0.img", "rb");

//si on a bien ouvert l'image
if (image != NULL)
{
	//on copie l'image
	fread(buffer, sizeof(unsigned short), taille, image);

	//on ferme le fichier
	fclose (image);
}
```

Fonction get:

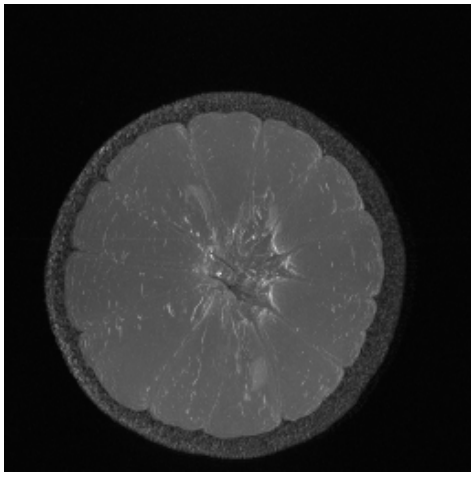
```
unsigned short getValue(unsigned short *buffer, int x, int y, int z)
{
	return buffer[(z * tailleX * tailleY) + ((tailleY - y - 1) * tailleX) + x ];
}

unsigned short getMin(unsigned short *buffer)
{
	unsigned short min = buffer[0];

	//on regarde tous les voxels de l'image
	for(int i = 0; i < (tailleX * tailleY * tailleZ); i++)
	{
		if(min > buffer[i])
			min = buffer[i];
	}
	return min;
}

unsigned short getMax(unsigned short *buffer)
{
	unsigned short max = buffer[0];

	//on regarde tous les voxels de l'image
	for(int i = 0; i < (tailleX * tailleY * tailleZ); i++)
	{
		if(max < buffer[i])
			max = buffer[i];
	}

	return max;
}
```

Fonction d’inversion d’octet:

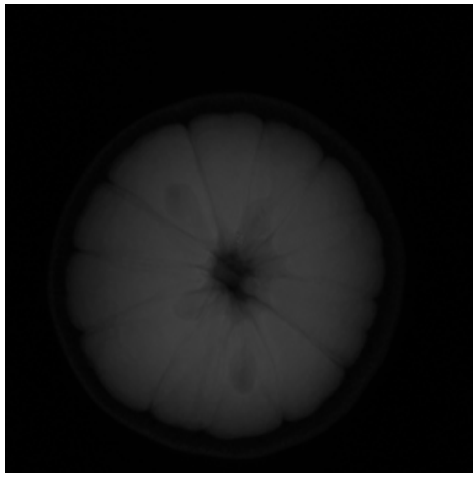
```
unsigned short inverserOctet(unsigned short octet)
{
	//merci google
	return (octet & 0xff) << 8 | ((octet & 0xff00) >> 8);
}
```

Exercice 4 à 6 :

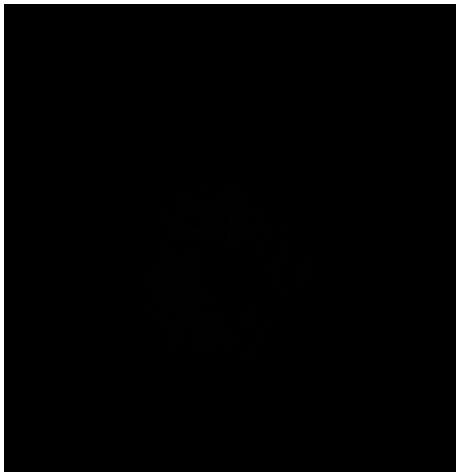
Orange:



Orange MIP

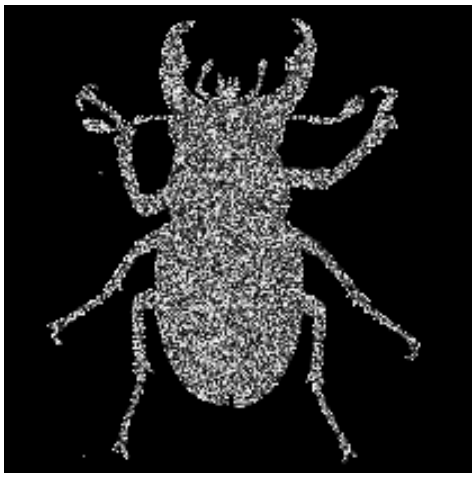


Orange AIP



Orange MinIP

Whatisit:



Whatisit MIP

Whatisit AIP

Whatisit MinIP

Fonction MIP, AIP, MinIP:

```
void MIP(unsigned short *buffer)
{
	int valeur;
	//height == hauteur
	//width == largeur
	ImageBase imOut(tailleX, tailleY, false);

	for(int i = 0; i < tailleX; i++)
	{
		for(int j = 0; j < tailleY; j++)
		{
			imOut[i][j] = 0;
			for(int k = 0; k < tailleZ; k++)
			{
				valeur = inverserOctet(getValue(buffer, i, j, k));
				if(valeur > imOut[i][j])
				{
					imOut[i][j] = valeur;
				}
			}
		}
	}
	imOut.save("MIP.pgm");
}
```

```
void AIP(unsigned short *buffer)
{
	int valeur;
	//height == hauteur
	//width == largeur
	ImageBase imOut(tailleX, tailleY, false);

	for(int i = 0; i < tailleX; i++)
	{
		for(int j = 0; j < tailleY; j++)
		{
			valeur = 0;
			for(int k = 0; k < tailleZ; k++)
			{
				valeur += inverserOctet(getValue(buffer, i, j, k));
			}

			imOut[i][j] = valeur / tailleZ;
		}
	}
	imOut.save("AIP.pgm");
}
```

```
void MinIP(unsigned short *buffer)
{
	int valeur;
	//height == hauteur
	//width == largeur
	ImageBase imOut(tailleX, tailleY, false);

	for(int i = 0; i < tailleX; i++)
	{
		for(int j = 0; j < tailleY; j++)
		{
			imOut[i][j] = 4000;
			for(int k = 0; k < tailleZ; k++)
			{
				valeur = inverserOctet(getValue(buffer, i, j, k));
				if((valeur < imOut[i][j]) && (valeur != 0))
				{
					imOut[i][j] = valeur;
				}
			}
		}
	}
	imOut.save("MinIP.pgm");
}
```