Lycée La Fayette
Champagne-sur-Seine • Fontaineroux

Rapport de Travaux Pratiques
BTS SN-IR

Matériaux OpenGL

Flavian Laxenaire
23 novembre 2021

Forme / 20 [coef. 1] →		Fond / 20 [coef. 2] →		Note / 20	
Qualité du rapport		Méthodologie			
Expression écrite		Respect du Cahier des Charges			
Pertinence de la rédaction		Qualité technique			
Respect des standards de codage		État d'avancement		Malus	

Sommaire

1. Crvba.h + Crvba.cpp

CRvba.h

```
#ifndef CRVBA
#define CRVBA
#include <string>
#include <iostream>
using namespace std;

class CRvba
{
private:
int rvba[4];

public:
Crvba() //initialisation de la couleur noire (0,0,0), sans transparence (255)
{
   for (int i = 0; i < 3; i++)
   rvba[i] = 0;
   rvba[3] = 255;
}
CRvba(int R, int V, int B, int A = 255) //constructeur acceptant 4 entiers, dont le dernier est optionnel (255)
{
   setRvba(R, V, B, A);
}
void setRvba(int R, int V, int B, int A = 255);
std::string printRvba() const;
void setR(int R); //modificateurs en externe
void setV(int V);
void setB(int B);
void setA(int A);
int getR() const { return rvba[0]; } //selecteurs inline
int getV() const { return rvba[1]; }
int getB() const { return rvba[2]; }
int getA() const { return rvba[3]; }
};
#endif
```

CRvba.cpp

```
#include "crvba.h"
#include <sstream>
#include <iomanip>
#include <iostream>

void CRvba::setRvba(int R, int V, int B, int A)
{
rvba[0] = R;
rvba[1] = V;
rvba[2] = B;
rvba[3] = A;
}

std::string CRvba::printRvba() const
{
std::ostringstream oss;
oss << std::setfill('0');
oss << "R:" << rvba[0];
oss << " V:" << rvba[1];
oss << " B:" << rvba[2];
oss << " A:" << rvba[3];
return oss.str();
}
```

```
void CRvba::setR(int R) this->rvba[0] = R;
void CRvba::setV(int V) this->rvba[1] = V;
void CRvba::setB(int B) this->rvba[2] = B;
void CRvba::setA(int A) this->rvba[3] = A;
```

main.cpp

```
/*****************************************/
/* Matériaux OpenGL TP2 */
/* 09/11/21 */
/* g++ -Wall -o main main.cpp crvba.cpp */
/* auteur : Flavian LAXENAIRE */
/*****************************************/
#include "crvba.h"
#include "cmateriau.h"
#include <iostream>
using namespace std;

int main(int argc, char **argv)
{
CRvba couleur1;
CRvba couleur2(115, 157, 172);
couleur1.setRvba(0, 255, 255);
cout << couleur2.printRvba() << endl;
cout << couleur1.printRvba() << endl;
return 0;
}
```

```
flavian-lxr@flavianlxr:~/Documents/bts snir 2nd annee/dev/cpp/2-RvbaMat$ g++ -Wall -o main main.cpp crvba.cpp
flavian-lxr@flavianlxr:~/Documents/bts snir 2nd annee/dev/cpp/2-RvbaMat$ ./main
R:115 V:157 B:172 A:255
R:0 V:255 B:255 A:255
```

2. Cmateriau.h + Cmateriau.cpp

Cmateriau.h

```
#ifndef CMATERIAU
#define CMATERIAU
#include <string>
#include <iostream>
#include "crvba.h"

class CMateriau
{
private:
std::string nom;  //nom couleur
int ambiante[4];  //réflexion ambiante
int diffuse[4];   //réflexion diffuse
int speculaire[4];//réflexion spéculaire
int brilliance;
int convert(int nombre) //fonction pour convertir un int 0..255 dans l'intervalle 0..1
{
if (nombre > 255)
nombre = 255;
else if (nombre < 0)
nombre = 0;
return (nombre / 255);
}
int undoConvert(int nombre)
{
if (nombre > 1)
nombre = 1;
else if (nombre < 0)
```

```
nombre = 0;
return (nombre * 255);
}

public:
Cmateriau() //constructeur par défault
{
setNom("défault");
setAmbiante(CRvba(0.2, 0.2, 0.2, 1.0));
setDiffuse(CRvba(0.8, 0.8, 0.8, 1.0));
setSpeculaire(CRvba(0.0, 0.0, 0.0, 1.0));
setBrilliance(0);
}
CMateriau(std::string nom, CRvba ambiante, CRvba diffuse, CRvba speculaire, int brilliance = 0) //constructeur pour
initialiser le materiau
{
setNom(nom);
setAmbiante(ambiante);
setDiffuse(diffuse);
setSpeculaire(speculaire);
setBrilliance(brilliance);
}
CMateriau(const CMateriau &c) //constructeur de copie
{
setNom(c.getNom());
setAmbiante(c.getAmbiante());
setDiffuse(c.getDiffuse());
setSpeculaire(c.getSpeculaire());
setBrilliance(c.getBrilliance());
}
void setNom(std::string nom);  //modificateurs
void setAmbiante(CRvba ambiante);
void setDiffuse(CRvba diffuse);
void setSpeculaire(CRvba speculaire);
void setBrilliance(int brilliance);
void setAlpha(int alpha);
CRvba getAmbiante() const;  //sélecteurs
CRvba getDiffuse() const;
CRvba getSpeculaire() const;
int getBrilliance() const;
std::string getNom() const { return nom; }
std::string printMateriau() const;  //l'affichage
std::string printAmbiante() const;
std::string printDiffuse() const;
std::string printSpeculaire() const;
};
#endif
```

cmateriau.cpp

```
#include "cmateriau.h"
#include <sstream>
#include <iomanip>
#include <iostream>

CRvba CMateriau::getAmbiante() const
{
CRvba ambiante2;
ambiante2.setRvba(this->ambiante[0], this->ambiante[1], this->ambiante[2], this->ambiante[3]);
return ambiante2;
}
CRvba CMateriau::getDiffuse() const
{
CRvba diffuse2;
diffuse2.setRvba(this->diffuse[0], this->diffuse[1], this->diffuse[2], this->diffuse[3]);
return diffuse2;
}

int CMateriau::getBrilliance() const
{
return this->brilliance;
}

CRvba CMateriau::getSpeculaire() const
{
CRvba speculaire2;
speculaire2.setRvba(this->speculaire[0], this->speculaire[1], this->speculaire[2], this->speculaire[3]);
return speculaire2;
}

void CMateriau::setNom(std::string nom)
{
this->nom = nom;
}
void CMateriau::setAmbiante(CRvba ambiante)
{
this->ambiante[0] = ambiante.getR();
this->ambiante[1] = ambiante.getV();
this->ambiante[2] = ambiante.getB();
this->ambiante[3] = ambiante.getA();
}
void CMateriau::setDiffuse(CRvba diffuse)
{
this->diffuse[0] = diffuse.getR();
this->diffuse[1] = diffuse.getV();
this->diffuse[2] = diffuse.getB();
this->diffuse[3] = diffuse.getA();
}
void CMateriau::setSpeculaire(CRvba speculaire)
{
this->speculaire[0] = speculaire.getR();
this->speculaire[1] = speculaire.getV();
this->speculaire[2] = speculaire.getB();
this->speculaire[3] = speculaire.getA();
}
void CMateriau::setBrilliance(int brilliance)
{
this->brilliance = brilliance;
}

void CMateriau::setAlpha(int alpha)
{
this->speculaire[3] = alpha;
this->diffuse[3] = alpha;
this->ambiante[3] = alpha;
```

```
}
std::string CMateriau::printAmbiante() const
{
std::ostringstream oss;
oss << std::setfill('0');
oss << "{" << ambiante[0] << "," << ambiante[1] << "," << ambiante[2] << "," << ambiante[3] << "}" << endl;
return oss.str();
}
std::string CMateriau::printDiffuse() const
{
std::ostringstream oss;
oss << std::setfill('0');
oss << "{" << diffuse[0] << "," << diffuse[1] << "," << diffuse[2] << "," << diffuse[3] << "}" << endl;
return oss.str();
}
std::string CMateriau::printSpeculaire() const
{
std::ostringstream oss;
oss << std::setfill('0');
oss << "{" << speculaire[0] << "," << speculaire[1] << "," << speculaire[2] << "," << speculaire[3] << "}" << endl;
return oss.str();
}
std::string CMateriau::printMateriau() const
{
std::ostringstream oss;
oss << std::setfill('0');
oss << "Couleur: " << nom << endl;
oss << "Réflexion ambiante: " << CMateriau::printAmbiante();
oss << "Réflexion diffuse: " << CMateriau::printDiffuse();
oss << "Réflexion spéculaire: " << CMateriau::printSpeculaire();
oss << "Brilliance: " << brilliance << endl;
return oss.str();
}
```

main2.cpp

```
#include "crvba.h"
#include "cmateriau.h"
#include <iostream>
using namespace std;

int main(int argc, char **argv)
{
CRvba couleur1;
CRvba couleur2(115, 157, 172);
couleur1.setRvba(0, 255, 255);
cout << couleur2.printRvba() << endl;
cout << couleur1.printRvba() << endl;
CMateriau mat1;
CMateriau mat2("turquoise", couleur2, CRvba(0.396, 0.741, 0.691), CRvba(0.297, 0.308, 0.306));
cout << mat2.printMateriau();
mat1 = mat2;
mat1.setAlpha(50);
cout << mat1.printMateriau();
return 0;
}
```

```
flavian-lxr@flavianlxr:~/Documents/bts snir 2nd annee/dev/cpp/2-RvbaMat$ g++ -Wall -o main main.cpp cmateriau.cpp crvba.cpp
flavian-lxr@flavianlxr:~/Documents/bts snir 2nd annee/dev/cpp/2-RvbaMat$ ./main
R:115 V:157 B:172 A:255
R:0 V:255 B:255 A:255
Couleur: turquoise
Réflexion ambiante: {115,157,172,255}
Réflexion diffuse: {0,0,0,255}
Réflexion spéculaire: {0,0,0,255}
Brilliance: 0
Couleur: turquoise
Réflexion ambiante: {115,157,172,50}
Réflexion diffuse: {0,0,0,50}
Réflexion spéculaire: {0,0,0,50}
Brilliance: 0
```

Q25. Finaliser l'ergonomie de la classe CMateriau en proposant des méthodes compatibles avec la primitive glMaterialfv() d'OpenGL qui ne peut accepter en argument qu'un pointeur sur un tableau de float (suffixe fv = foat vector).

Cmateriau.h

```
/**/
    float* rAmbiante() const;
    float* rDiffuse() const;
    float* rSpeculaire() const;
    float* Brilliance();
/**/
```

Cmateriau.cpp

```
float *Cmateriau::rAmbiante() const
{
    return (float *)ambiante;
}
float *CMateriau::rDiffuse() const
{
    return (float *)diffuse;
}
float *CMateriau::rSpeculaire() const
{
    return (float *)speculaire;
}
float *CMateriau::Brilliance()
{
    return &brilliance;
}
```