Coding Standard - Conventions de codage
Programmation C++

Merci de respecter les conventions suivantes pour l'écriture de vos classes C++.
Si possible, préférez la langue anglo-saxonne pour les noms de fonctions, de classes ou de variables utilisées dans vos programmes.
Enfin, plutôt que les types standards (int, float, double, ...) définis dans le langage C, utilisez ceux définis dans le fichier Types.h (disponible sous Moodle) pour faciliter la portabilité de vos programmes.

1. Les noms

Noms de classes :

- Le nom d'une classe doit être celui d'une entité abstraite ou physique.
  Exemples : class Camera, class Quaternion, class AnimatedMesh...
- La première lettre d'un **nom de classe** doit être une **majuscule**.
- Utilisez les lettres majuscules pour séparer les mots, les lettres minuscules pour le reste d'un mot. Pas d'underscore.
- Exemple : class QuaternionBasedCamera

Noms des fonctions membres d'une classe :

- Le nom d'une fonction doit être un verbe.
- La première lettre d'un **nom de fonction membre** doit être une **minuscule**.
- Utilisez les lettres majuscules pour séparer les mots, les lettres minuscules pour le reste d'un mot. Pas d'underscore.
- Les préfixes précisent l'utilisation de la fonction :
  - get : récupère une valeur
  - set : initialise une variable
  - is ou has : effectue un test
- Eviter de repréciser le nom de la classe dans le nom d'une fonction
- Exemples : void move(...), bool hasMoved() (plutôt que bool hasCameraMoved() si cette fonction est dans l'interface de la classe Camera), ...

Noms des fonctions C

- Le nom d'une fonction C ne doit comporter que des minuscules.
- Les mots sont séparés par un underscore.
- Exemples : void init_render_device(...), int32 get_render_device_status(), ...

Noms des variables membres d'une classe :

- Le nom d'une variable membre d'une classe doit commencer par m_
- Utilisez les lettres majuscules pour séparer les mots, les lettres minuscules pour le reste d'un mot. Pas d'underscore pour séparer les mots.

Exemples : float32 m_X;
float32 m_PrimaryColor;

Noms des variables locales à une fonction :

- Le nom d'une variable locale ne doit comporter que des minuscules.
- Utilisez les underscores pour séparer les mots.
  Exemples : int32 x;
  float32* tab_colors;

Noms des variables globales :

- Le nom d'une variable globale doit commencer par g_.
- Utilisez les lettres majuscules pour séparer les mots, les lettres minuscules pour le reste d'un mot. Pas d'underscore pour séparer les mots.
- Exemples : uint32 g_NbrMeshes;
  uint32 g_Timer;

Noms des constantes :

- Le nom d'une constante ne doit comporter que des majuscules.
- Les mots sont séparés par un underscore.
- Exemple : const float32 HALF_PI = 3.14156f/2.0f;

Noms de types :

- Même convention que pour les classes.
- Ajouter *Type* à la fin du type déclaré pour les types déclarés dans des classes.
- Exemples : typedef unsigned int uint32;
  typedef float float32;
  typedef double float64;
  typedef std::list<Camera> CameraListType;

Noms des valeurs des types énumérés :

- Même convention que pour les constantes
- Exemple :
```
enum LightType
{
    DIRECTIONAL_LIGHT    = 0,
    POINT_LIGHT,
    SPOT_LIGHT
};
```

Noms des headers et fichiers d'implémentation

- Même nom que la classe associée, par exemple Quaternion.h et Quaternion.cpp.
- Pour les fichiers d'implémentation, terminez par l'extension cpp.
- Pour les headers, insérez des lignes de garde pour éviter l'inclusion multiple.
  Exemple :

```
#ifndef __CLASSNAME__
#define __CLASSNAME__
```

```
    ...
#endif
```

2. Déclaration de variables et pointeurs

- La déclaration de plusieurs variables de même type doit être réalisée sur plusieurs lignes, sauf si les variables sont utilisées dans le même contexte.
  Exemples : int32 i, j, k;
  float32 m_X, m_Y, m_Z;

  MAIS :

  uint32 m_NbrVertices; // nombre de vertices du mesh
  uint32 m_NbrTextures; // nomre de textures utilisées par le mesh
  Cela permet ainsi d'adjoindre éventuellement des commentaires aux variables.
- Les pointeurs doivent être indiqués immédiatement après le type de la variable.
  Exemple : uint32* tab_indices;
- Si plusieurs variables sont des pointeurs de même type, les déclarer ligne par ligne.
  Exemples : float32* tab_vertices;
  float32* tab_colors;

3. Structures conditionnelles et boucles

Indentez les structures conditionnelles et les boucles comme suit :

```
if ( ... )
{
  ...
}
else if ( ... )
{
  ...
}
else
{
  ...
}


switch ( ... )
{
  case 1:
    ...
    break;

  case 2:
    ...
    break;

  ...
}
```

```
while ( ... )
{
  ...
}


do
{
  ...
}
while ( ... );
```

Laissez un espace entre le mot clé de structure conditionnelle ou de boucle et la parenthèse ouvrante. Ce ne sont pas des fonctions...

4. Commentaires

Commentez autant que possible votre code. Utilisez les commentaires suivants pour distinguer les différentes sections d'une classe dans un fichier d'implémentation :

```
#include "XX.h"  // classe implémentée

...

//////////////////////////////// PUBLIC /////////////////////////////////////

//============================== LIFECYCLE ====================================

XX::XX()
{
  ...
}

XX::XX(const XX&)
{
  ...
}

XX::~XX()
{
  ...
}


//============================== OPERATORS ====================================

XX& XX::operator=(const XX&);
{
  ...
}


//============================== OPERATIONS ===================================

void XX::translate( float x, float y, float z )
{
  ...
}
```

```
void XX::rotate( float angle, float x, float y , float z )
{
  ...
}


//============================= ATTRIBUTE ACCESSORS ==========================

float XX::getX()
{
  ...
}

void XX::setX( float x )
{
  ...
}


//============================= INQUIRY    ====================================

bool XX::isTransparent() const
{
  ...
}


/////////////////////////////// PROTECTED  ///////////////////////////////////
...


/////////////////////////////// PRIVATE    ///////////////////////////////////
...
```

Les commentaires propres à une fonction devront respecter le formalisme suivant :

```
/**************************************************************************
  * Name:
  * Description:
  * Inputs:
    - parameter1: description of parameter1
    ...
    - parameterN: description of parameterN
  * Returns:
    - value: description of the returned value
**************************************************************************/
```

5. Taille maximum d'une ligne

N'importe quelle ligne d'un programme C++ ne devrait pas comporter plus de 80 caractères pour pouvoir être visible d'un coup d'oeil.