Déclaration de types

Using / Typedef

```
using uid = long; // > C++11

typedef long uid; // < C++11

uid myId = 0;
```

Les mots-clés `using` et `typedef` permettent de créer un synonyme à un type existant.

Using / Typedef

```
using v3f = float[3]; // > C++11

typedef float v3f[3]; // < C++11

v3f position;
position[0] = 1.5f;
```

Il est possible de définir des synonymes de tableaux de taille fixe.

Using / Typedef

```
using uid = long;
uid myId = 0;
int i = myId / 2;
```

Les opérations et les casts définis pour le type de base le sont également pour le synonyme.

Using / Typedef

Les synonymes sont utilisés pour :

- Simplifier la syntaxe.
- Clarifier le cadre d'utilisation d'un type.

Enum

```
enum STATE {
    IDLE,
    MOVING,
};
STATE myState = STATE::MOVING;
```

Le mot-clé `enum` permet de définir un type associé à une liste discrète de valeurs.

Enum

```
STATE myState = STATE::MOVING;
if (myState == STATE::MOVING) {
...
}
```

Les valeurs de l'enum peuvent être utilisées comme n'importe quelles autres valeurs dans le code.

Enum

```
enum STATE {
  IDLE = 0,
  MOVING = 5,
};
STATE myState = (STATE)5; // => STATE::MOVING
```

Une correspondance à une **valeur entière** est associée **par défaut** à chaque valeur de l'enum : 0, 1, 2, ...

Il est possible de **redéfinir ces valeurs**.

Enum

`enum` est utilisé pour définir des types à états finis.

On l'utilisera entre autre pour lister un ensemble de possibilités (**événements**) ou d'états (**machines à états**).

Struct

```
struct Human {
  std::string name;
  int age;
  float size;
};

Human bobby;
bobby.name = "bobby";
bobby.age = 20;
bobby.size = 1.7f;
```

Une **structure** est une variable regroupant plusieurs sous-variables.

Struct

```
struct Human {
  std::string name;
};

struct Alien {
  std::string name;
};

Human bobby;
Alien et = bobby; // Erreur !
```

Il n'existe pas de **cast** possible entre **structures**, même si elles portent les mêmes données.

Struct

```
struct Human {
  std::string name;
  float age;
  float size;
};

Human bobby = {"bobby", 20.2f, 1.7f};
```

Il est possible d'initialiser les variables de type **structure** avec { }.

Attention cependant au changement d'ordre lors des modifications sur la **structure**.

Struct

```
struct Human {
  std::string name = "NA";
  float age = 0f;
  float size = 0.5f;
};

Human bobby; // Initialised to {"NA", 0f, 0.5f}
```

Il est possible de donner des valeurs par défaut à une structure.