虚幻4高级程序-第三阶段02

UNREAL 4 HIGH INTEGRATION PROGRAM DEVELOPMENT ENGINEER

01 UE基本数据类型

UEC++基本数据类型

为了方便跨平台，UE对于C++基本数据类型进行深度重定义，方便平台扩展特性，增加UE的移植便捷性。

禁止在UE中使用C++的基本数据类型，这样会影响引擎的跨平台特性

- bool 代表布尔值 (永远不要假设布尔值的大小) 。BOOL 将不会进行编译。
- TCHAR 代表字符型（永远不要假设TCHAR的大小）。
- uint8 代表无符号字节（占1个字节）。
- int8 代表有符号的字节（占1个字节）。
- uint16 代表无符号"短整型" (占2 个字节)。
- int16 代表有符号"短整型" (占2 个字节)。
- uint32 代表无符号整型（占4字节）。
- int32 代表带符号整型（占4字节）。
- uint64 代表无符号"四字" (8个字节)。
- int64 代表有符号"四字"（8个字节）。UE
- float 代表单精度浮点型 (占4 个字节)。
- double 代表双精度浮点型 (占8 个字节)。
- PTRINT一个符号整数和一个指针一样大小（用来标记指针的大小） (永远不要假设PTRINT的大小)。

02 字符编码

编码

编码是信息从一种形式或格式转换为另一种形式的过程，也称为计算机编程语言的代码简称编码。用预先规定的方法将文字、数字或其它对象编成数码，或将信息、数据转换成规定的电脉冲信号。编码在电子计算机、电视、遥控和通讯等方面广泛使用。编码是信息从一种形式或格式转换为另一种形式的过程。解码，是编码的逆过程。

编码的故事

在程序设计中，编码是不能被抛弃的话题。计算机本身对于文本是无法识别的！对于计算机来说，他只有简单的电子信号（0和1），通过组合构成了二进制数据。而对于人来说我们不光要认知数字，还需要认知文本！但是文本在计算机中就无法被直接储存了！那么人们想到了将文本与密码（编码）。用特定的密码，代表特定的文本，聪明的人们用这种办法解决了计算机无法储存文本的问题！

但是！由于世界上有很多种国家，他们使用的文本类型各种各样，这样又出现了问题，早起的密码本不够用了，他只能翻译英文的字符和一些少量的字符（ANSII编码），这对于亚洲国家地区来说是非常痛苦的！亚洲国家的字符量非常庞大，对于计算机来说，如果他只有ANSI编码，这明显不够！为了解决这个问题。人们构建出了更多的编码集，例如GBK，国标，UTF，ISO，BIG5，HZ码等。

这有出问题了，因为我们经常遇到一种情况，当两个计算机系统编码是相同的，那么我们之间可以友好的沟通，但是如果不同！那就会出现非常诡异的事情，例如乱码！这也令人非常头痛！

转码，专门为了解决编码之间的差异问题！你可以理解为是翻译器！

注意编码解决的是文本问题。

文本格式

文本在进行储存时会选择文本的格式。文本常用表示格式分为二进制（无格式），文本（ASCII），UTF-8，UTF-16

虚幻引擎4（UE4）中的所有字符串都作为FStrings或TCHAR数组以UTF-16 格式存储在内存中。大多数代码假设2个字节等于一个代码点，因此只支持基本多文种平面（BMP），这样虚幻内部编码可以更准确地描述为UCS-2。字符串以适合于当前平台的字节次序存储。

当向从磁盘序列化到程序包，或在联网期间序列化时，TCHAR字符小于0xff的字符串均存储为一串8位字节，否则存储为双字节UTF-16字符串。序列化代码可以根据需要处理任何字节次序转换。

03 UE4中的字符类型

字符类型

UE4中提供多种字符类型进行处理数据，在不同的情景下，我们需要选择不同的类型进行操作。

区别：大小不同，编码方式不同，所有的文本在进行存储的时候，编译器编译阶段会根据编码类型进行转码。

```
//~ Character types.
/// An ANSI character. Normally a signed type.
typedef FPlatformTypes::ANSICHAR    ANSICHAR;
/// A wide character. Normally a signed type.
typedef FPlatformTypes::WIDECHAR    WIDECHAR;
/// Either ANSICHAR or WIDECHAR, depending on whether the platform supports wide characters or the requirements of the licensee.
typedef FPlatformTypes::TCHAR       TCHAR;
/// An 8-bit character containing a UTF8 (Unicode, 8-bit, variable-width) code unit.
typedef FPlatformTypes::CHAR8       UTF8CHAR;
/// A 16-bit character containing a UCS2 (Unicode, 16-bit, fixed-width) code unit, used for compatibility with 'Windows TCHAR' across multiple platforms.
typedef FPlatformTypes::CHAR16      UCS2CHAR;
/// A 16-bit character containing a UTF16 (Unicode, 16-bit, variable-width) code unit.
typedef FPlatformTypes::CHAR16      UTF16CHAR;
/// A 32-bit character containing a UTF32 (Unicode, 32-bit, fixed-width) code unit.
typedef FPlatformTypes::CHAR32      UTF32CHAR;
```

转码宏

有些时候，我们需要主动的进行转码操作，这也是很有必要的，虚幻中提供了一些转码操作宏，来帮助我们进行转码操作。

```
// Usage of these should be replaced with StringCasts.
#define TCHAR_TO_ANSI(str) (ANSICHAR*)StringCast<ANSICHAR>(static_cast<const TCHAR*>(str)).Get()
#define ANSI_TO_TCHAR(str) (TCHAR*)StringCast<TCHAR>(static_cast<const ANSICHAR*>(str)).Get()
#define TCHAR_TO_UTF8(str) (ANSICHAR*)FTCHARToUTF8((const TCHAR*)str).Get()
#define UTF8_TO_TCHAR(str) (TCHAR*)FUTF8ToTCHAR((const ANSICHAR*)str).Get()
```

04 对象字符串

UE中的对象字符串

1. FName，资源命名字符串，FName 通过一个轻型系统使用字符串。在此系统中，特定字符串即使会被重复使用，在数据表中也只存储一次。FNames 不区分大小写（大小写不是他比较的依据）。它们为不可变，无法被操作。
   FNames 的存储系统和静态特性决定了通过键进行 FNames 的查找和访问速度较快。FName 子系统的另一个功能是使用散列表为 FName 转换提供快速字符串。
2. FText表示一个显示字符串，用户的显式文本都需要由FText进行处理。支持格式化文本，不提供修改函数，无法进行内容修改
3. FString，可以被操作的字符串。开销大于其他类字符串类型

05 FString

06 FName

07 FText

火星时代教育

感谢聆听

THANK YOU

帮助更多的人实现梦想