

Debuggage

Sylvain GROSDÉMOUGE



Introduction

Rappel

Variables

Zone mémoire

Registres

Pile d'appel

Pointeur d'instruction

Debugger

Utilisation

Exemple concret

Bonnes pratiques

Variables

Zone mémoire associée à un nom, et un type.

Par exemple :

```
int          a;          // Variable de type entier.  
char *       ptr;        // Variable de type pointeur sur char.  
CObject      object;     // Variable de type CObject.
```

Introduction

Rappel

Variables

Zone mémoire

Registres

Pile d'appel

Pointeur d'instruction

Debugger

Utilisation

Exemple concret

Bonnes pratiques

Zone mémoire

Définie par une adresse de départ dans la mémoire, et une taille :

```
char szBuffer[256]
```

définit une zone mémoire de 256 octets

```
char * pBuffer = new char[1024];
```

alloue une zone mémoire de 1024 octets

pBuffer et szBuffer sont des variables qui contiennent les adresses de départ des deux zones.

Introduction

Rappel

Variables

Zone mémoire

Registres

Pile d'appel

Pointeur d'instruction

Debugger

Utilisation

Exemple concret

Bonnes pratiques

Registres

Emplacement de mémoire interne à un processeur.

Plusieurs types :

- Registres entiers (sur x86, EAX, EBX, ...)
- Registres flottants (sur x87, fp1, fp2, ...)
- Registres d'adressage (CS, DS, ES, ...)
- Registres d'index (SI, DI, ...)

- Registre(s) d'états (flags) (sur X86, FLAGS)

- Pointeur d'instructions (sur x86, IP)

- Pointeur de pile (sur x86, SP)

Introduction

- Rappel

 - Variables

 - Zone mémoire

 - Registres

 - Pile d'appel

 - Pointeur d'instruction

- Debugger

Utilisation

Exemple concret

Bonnes pratiques

Pile d'appel

Permet de stocker, sous la forme d'une pile, la trace de l'endroit où chaque fonction active doit retourner à la fin de son exécution.

Utilisée pour connaître la liste des fonctions appelées successivement avant d'arriver à l'instruction courante.

Introduction

- Rappel

 - Variables

 - Zone mémoire

 - Registres

 - Pile d'appel

 - Pointeur d'instruction

- Debugger

Utilisation

Exemple concret

Bonnes pratiques

Pointeur d'instruction

Registre qui contient l'adresse mémoire de l'instruction en cours d'exécution, ou prochainement exécutée (selon l'architecture).

Une fois que le processeur a mis l'instruction courante en état d'exécution, il est automatiquement incrémenté.

Introduction

Rappel

Variables

Zone mémoire

Registres

Pile d'appel

Pointeur d'instruction

Debugger

Utilisation

Exemple concret

Bonnes pratiques

Debugger (ou débogueur)

« Logiciel qui aide un développeur à trouver les bugs présents dans un programme »

Il permet notamment :

- de stopper l'exécution d'un programme à un moment donné
- de l'exécuter pas à pas
- de contrôler son état à tout instant (variables, mémoires, ...)

Phase de débogage

Le debuggage se déroule généralement en trois étapes :

- Arrêt de l'exécution
- Trace de l'exécution pas à pas
- Observation de l'état des différents composants

Introduction

Utilisation

- Phase de debuggage
 - Arrêt de l'exécution
 - Break / Continue
 - Breakpoint
 - Conditional breakpoint
 - Watchpoint
 - Trace de l'exécution
 - Observation de l'état

Exemple concret

Bonnes pratiques

Break / Continue

Permet de stopper l'exécution d'un programme en cours d'exécution, par une commande (ligne de commande) ou l'appui sur un bouton (IDE).

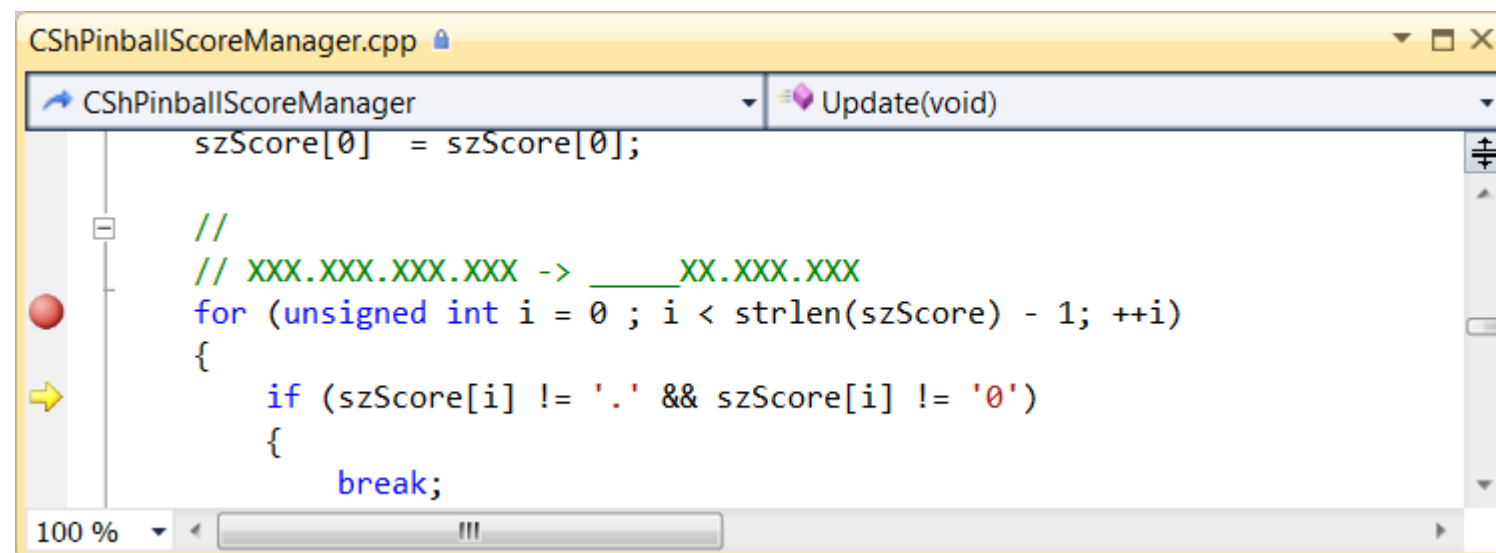
Le debugger se positionne alors sur la ligne correspondant à la valeur du pointeur d'instruction lors du break.

La commande continue permet de poursuivre l'exécution du programme.

Breakpoint

Permet de stopper l'exécution du programme lorsque le pointeur d'instruction arrive l'adresse associée au breakpoint.

Dans la pratique, on observe une 'pastille' à côté de la ligne concernée dans le code.



Conditional Breakpoint

Permet d'arrêter l'exécution d'un programme en fonction d'une condition sur une variable ou une zone mémoire.

Par exemple :

- Stopper le programme lorsque la variable 'iVariable' est > 32

Introduction

Utilisation

Phase de debuggage

- Arrêt de l'exécution

 - Break / Continue

 - Breakpoint

 - Conditional breakpoint

 - Watchpoint

- Trace de l'exécution

- Observation de l'état

Exemple concret

Bonnes pratiques

Watchpoint

Permet d'arrêter l'exécution d'un programme en fonction d'un changement constaté sur une zone mémoire.

Par exemple, stopper le programme quand le contenu présent à l'adresse mémoire 0x45D80010.

Introduction

Utilisation

Phase de debuggage

- Arrêt de l'exécution

- Trace de l'exécution

 - Step over

 - Step into

 - Step out

- Observation de l'état

Exemple concret

Bonnes pratiques

Step over

(F10 sous Visual Studio)

Exécute la ligne courante, sans entrer à l'intérieur des fonctions appelées

Introduction

Utilisation

Phase de debuggage

- Arrêt de l'exécution

- Trace de l'exécution

 - Step over

 - Step into

 - Step out

- Observation de l'état

Exemple concret

Bonnes pratiques

Step into

(F11 sous Visual Studio)

Exécute un pas, en entrant dans les fonctions appelées sur la ligne courante par ordre d'exécution

Introduction

Utilisation

Phase de debuggage

Arrêt de l'exécution

Trace de l'exécution

Step over

Step into

Step out

Observation de l'état

Exemple concret

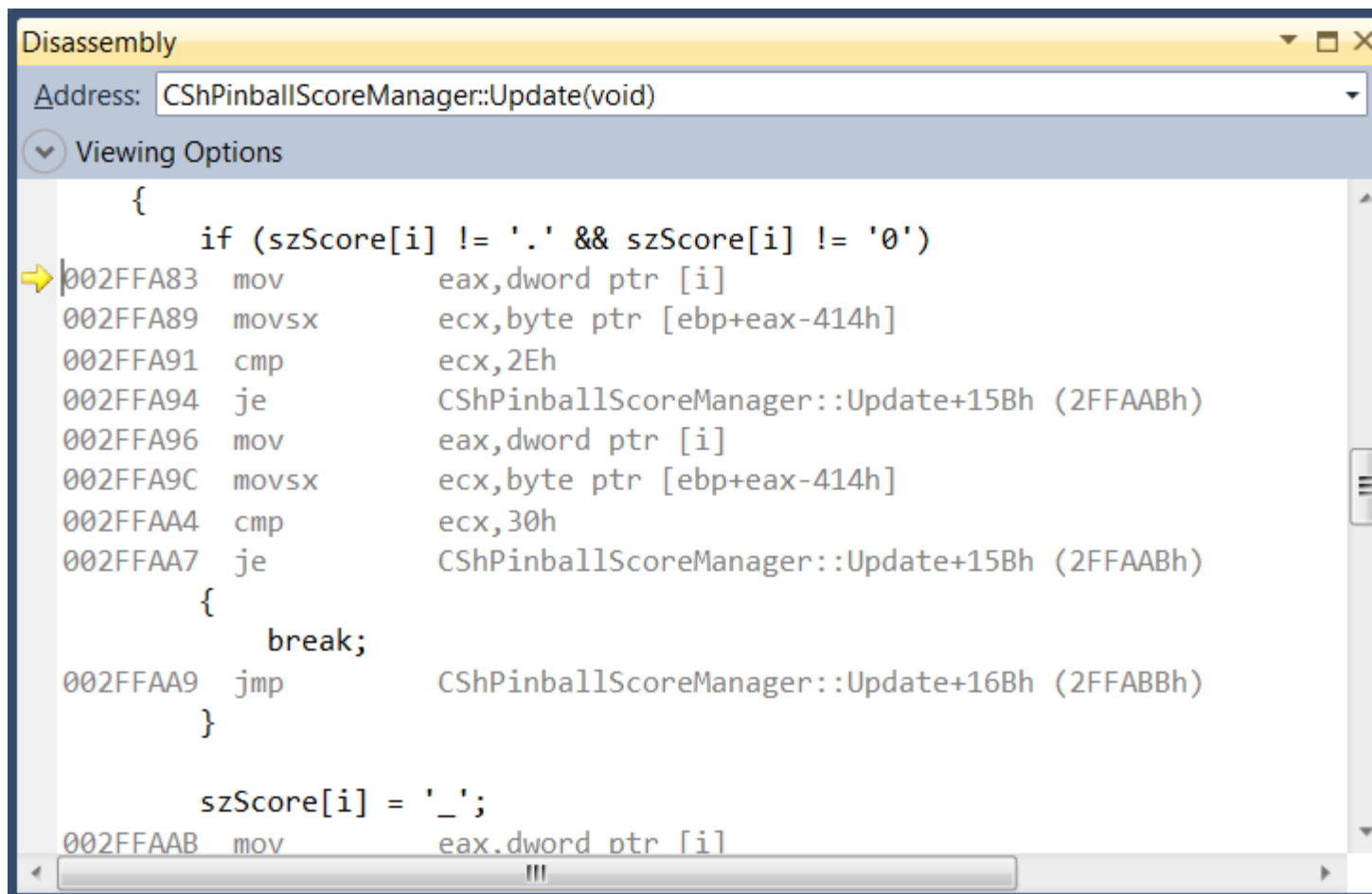
Bonnes pratiques

Step out

(Shift + F11 sous Visual Studio)

Exécute le programme jusqu'à la sortie de la fonction courant.

Switch source code / désassembleur



```
Disassembly
Address: CShPinballScoreManager::Update(void)
Viewing Options
{
    if (szScore[i] != '.' && szScore[i] != '0')
    002FFA83 mov     eax,dword ptr [i]
    002FFA89 movsx   ecx,byte ptr [ebp+eax-414h]
    002FFA91 cmp     ecx,2Eh
    002FFA94 je      CShPinballScoreManager::Update+15Bh (2FFAABh)
    002FFA96 mov     eax,dword ptr [i]
    002FFA9C movsx   ecx,byte ptr [ebp+eax-414h]
    002FFAA4 cmp     ecx,30h
    002FFAA7 je      CShPinballScoreManager::Update+15Bh (2FFAABh)
    {
        break;
    002FFAA9 jmp     CShPinballScoreManager::Update+16Bh (2FFABBh)
    }

    szScore[i] = '_';
    002FFAAB mov     eax,dword ptr [i]
```


Introduction

Utilisation

Phase de debuggage

- Arrêt de l'exécution

- Trace de l'exécution

- Observation de l'état

 - Switch source / asm

 - Variables

 - Locals

 - Watch

 - Zone mémoire

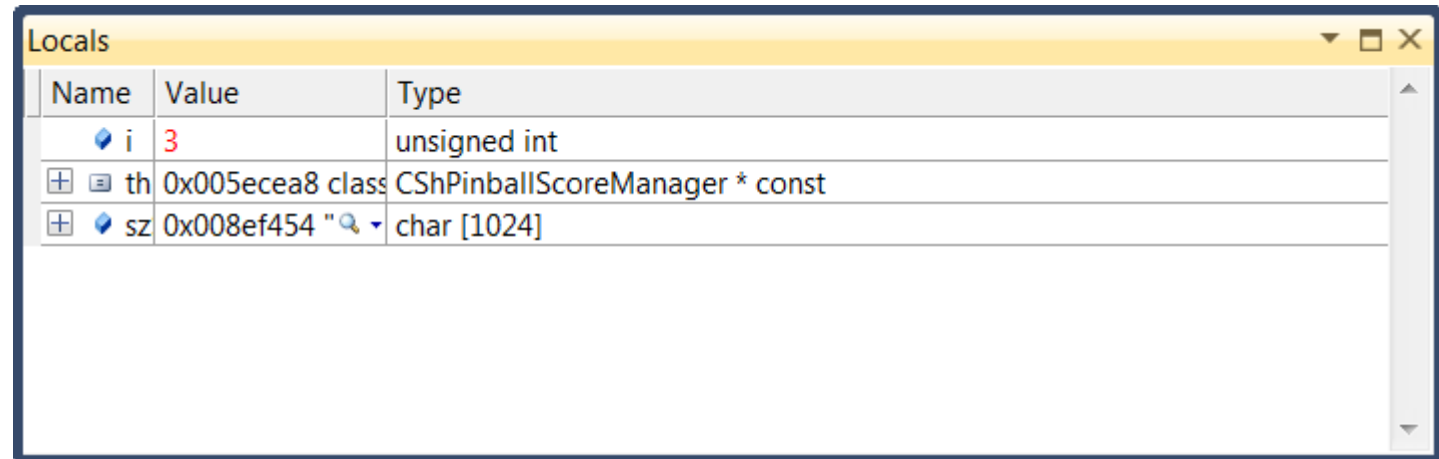
 - Registres

 - Pile d'appel (call stack)

Exemple concret

Bonnes pratiques

Variables - Locals



The screenshot shows a debugger window titled 'Locals'. It contains a table with three columns: 'Name', 'Value', and 'Type'. The table lists three local variables: 'i' with value 3 and type 'unsigned int'; 'th' with value '0x005ecea8 class CShPinballScoreManager * const'; and 'sz' with value '0x008ef454' and type 'char [1024]'. Each row has a small icon to its left, and the 'th' and 'sz' rows have expandable icons.

Name	Value	Type
i	3	unsigned int
th	0x005ecea8 class CShPinballScoreManager * const	
sz	0x008ef454 " " char [1024]	

Introduction

Utilisation

Phase de debuggage

Arrêt de l'exécution

Trace de l'exécution

Observation de l'état

Switch source / asm

Variables

Locals

Watch

Zone mémoire

Registres

Pile d'appel (call stack)

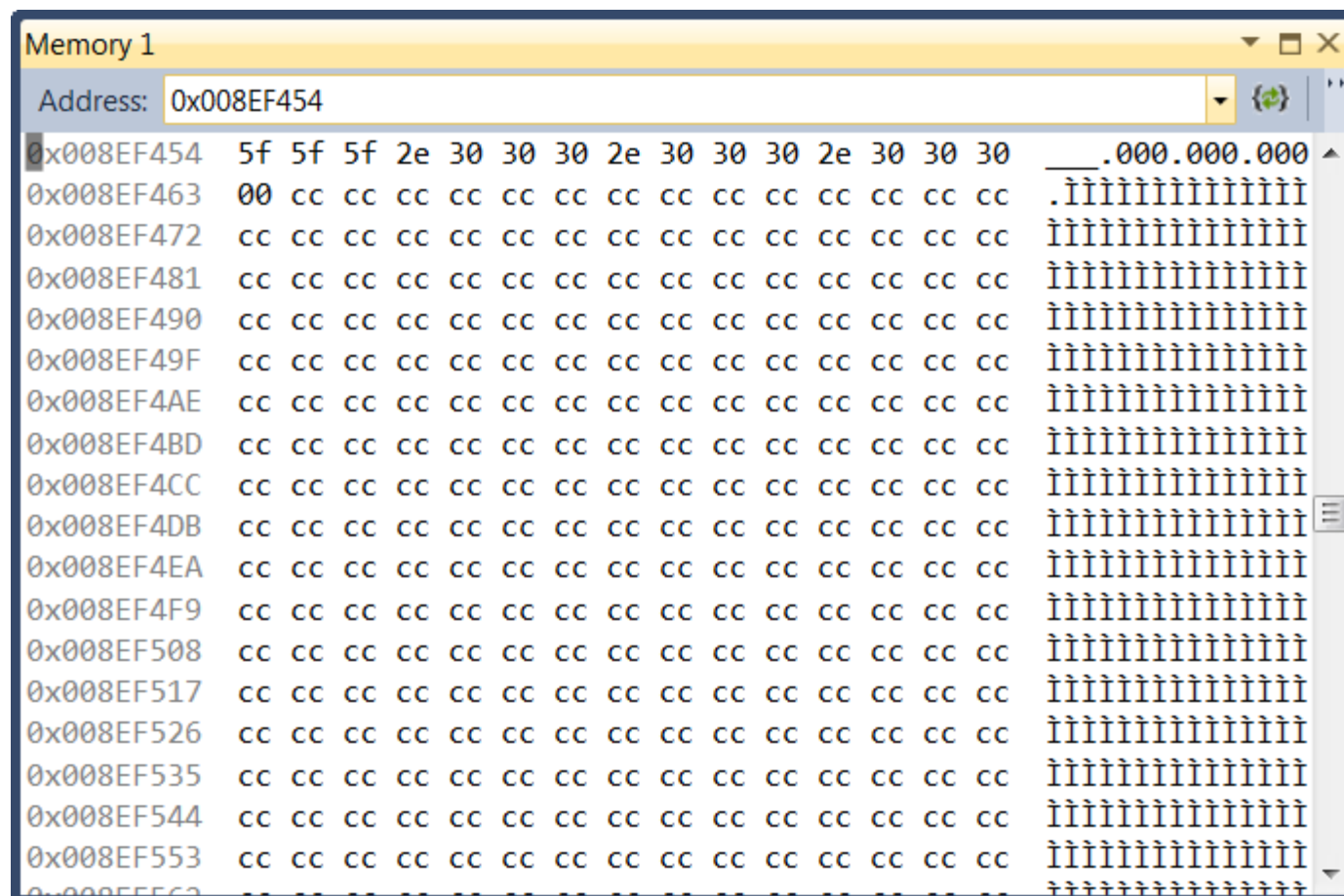
Exemple concret

Bonnes pratiques

Variables - Watch

Watch 1		
Name	Value	Type
<input type="checkbox"/> szScore	0x008ef454 "____,000.000.000"	char [1024]
<input type="checkbox"/> szScore[1]	95 '_'	char
<input type="checkbox"/> szScore[2]	95 '_'	char
<input type="checkbox"/> szScore[3]	46 '.'	char
<input type="checkbox"/> szScore[3]	46 '.'	char
<input type="checkbox"/> this	0x005ecea8 class CShPinballScoreM	CShPinballScoreManager * const
<input type="checkbox"/> &((*this).m_bSpecialTable)	0x005eceba	bool *

Zone mémoire



Introduction

Utilisation

Phase de debuggage

- Arrêt de l'exécution

- Trace de l'exécution

- Observation de l'état

 - Switch source / asm

 - Variables

 - Zone mémoire

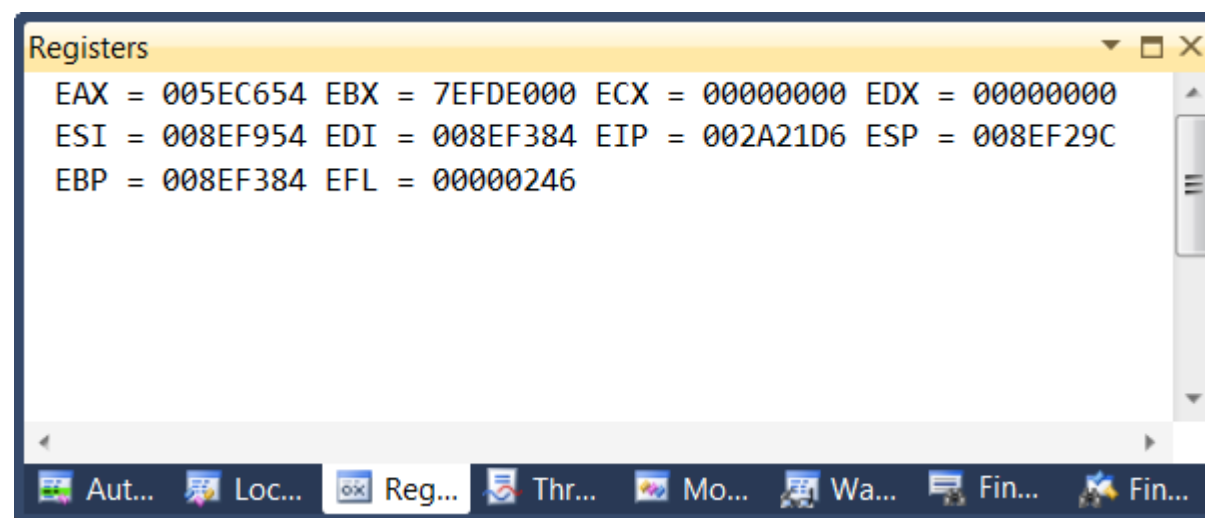
 - Registres

 - Pile d'appel (call stack)

Exemple concret

Bonnes pratiques

Registres



Introduction

Utilisation

Phase de debuggage

Arrêt de l'exécution

Trace de l'exécution

Observation de l'état

Switch source / asm

Variables

Zone mémoire

Registres

Pile d'appel (call stack)

Exemple concret

Bonnes pratiques

Pile d'appel (Call Stack)

Call Stack		
Name	Language	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CShPinballBackground::Update(float dt) L	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CShPinballGame::Update(float deltaTimeIr	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CShPinballGameStateGame::Update(float d	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CShPinballGameStateManager::Update(flo	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CMultiResolutionApplication::OnPreUpdat	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!ShApplication::OnPreUpdate(float deltaTir	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!ShApplication::OnPreUpdate(float deltaTir	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CSnApplication::update() Line 2159 + 0x1	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!CShApplication::MainLoop() Line 1270 + 0	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!ShMain(void * pUserData, const ShDisplay	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!WinMain(HINSTANCE__ * hInstance, HINST	C++	
QuanticPinballLite_2010_PC_Win32_Debug.exe!_tmainCRTStartup() Line 547 + 0x2c byte:	C	
QuanticPinballLite_2010_PC_Win32_Debug.exe!WinMainCRTStartup() Line 371	C	
kernel32.dll!76cc336a()		
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]		
ntdll.dll!77d89f72()		
ntdll.dll!77d89f45()		

Exemple concret

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Coding Standard

- Uniformiser le code
- Simplifier la lecture, la maintenance et le debuggage
- Se prémunir de certains bugs en respectant certaines règles
- Permettre plus facilement de mettre en place des tests unitaires

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Constructeurs :

```
CObject
: m_variable1()
, m_variable2()
, m_variable3()
{

}
```

plutôt que :

```
CObject
: m_variable1(),
  m_variable2(),
  m_variable3()
{

}
```

Plus lisible pour commenter une variable :

```
CObject:: CObject(void)
: m_variable1()
, m_variable2()
//, m_variable3()
```

plutôt que :

```
CObject:: CObject(void)
: m_variable1(),
  m_variable2()/*,
  m_variable3()*/
```


Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Indentation :

```
for (int i = 0; i < count ; ++i)
{
    // ...
}
```

plutôt que :

```
for (int i = 0; i < count ; ++i) {
    // ...
}
```

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Constructeurs :

```
Class CObject
{
    explicit      CObject      (... )
    virtual      ~CObject      (void) ;

    // ...

};
```

Utiliser des constructeurs explicites
et destructeurs virtuels dans tous les
cas !

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Constructeurs :

```
Class CObject  
{
```

```
    ...
```

```
private:
```

```
    char *  
    int
```

```
    m_ptr1;  
    m_intVar;
```

```
};
```

```
CObject::CObject  
: m_ptr1(NULL)  
, m_intVar(2)  
{  
  
}
```

Vous n'êtes pas garantis que le gestionnaire de mémoire effectue un memset 0 sur chaque allocation !

(donc possibilité de mémoire non initialisée, donc au contenu aléatoire si l'initialisation n'est pas faite dans le constructeur)

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Utiliser SAFE_FREE / SAFE_DELETE / ... :

```
#define SH_SAFE_DELETE_ARRAY(ptr) \
if (shNULL_!= ptr) \
{ \
    delete [] ptr; \
    ptr = shNULL; \
}
```

```
#define SH_SAFE_DELETE(ptr) \
if (shNULL_!= ptr) \
{ \
    delete ptr; \
    ptr = shNULL; \
}
```

```
#define SH_SAFE_FREE(ptr) \
if (shNULL_!= ptr) \
{ \
    SH_FREE(ptr); \
    ptr = shNULL; \
}
```

Evite d'utiliser un pointeur qui pointe sur un contenu détruit.

(Le debugger stoppe sur les ptr-> quand ptr == NULL, pas si ptr contient une adresse corrompue)

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Utiliser des ASSERT :

```
#include <assert.h>
```

```
assert(NULL != ptr);
```

Stoppe le debugger si la condition passée en argument de la fonction `assert(...)` n'est pas valide.

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Nom des variables :

Préfixe des class par un CSh dans les librairies internes, et C dans les projets (limite les conflits de namespaces)

Variables membres d'une class commencent par m_ (identification rapide des variables membres par rapport aux variables locales)

Variables membres d'une class, de type pointer, commencent par m_p (identification rapide des pointeurs)

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Nom des méthodes :

Commencent par une majuscule, puis majuscule à chaque mot.

Evite les duplicatas avec des casses différentes, et les potentiels problèmes induits (appel de la mauvaise méthode).

Introduction

Utilisation

Exemple concret

Bonnes pratiques

Coding standard

Intérêt

Exemples de cas

Simplifier la lecture

Prémunir certains bugs

Assignation :

```
if (NULL != ptr)
{
    // ...
}
```

plutôt que :

```
if (ptr != NULL)
{
    // ...
}
```

En cas de faute de frappe :

```
If (NULL = ptr)
    → Ne compile pas !
```

```
If (ptr = NULL)
    → Compile !
```


Enumérations :

```
enum ELevel
{
    e_level_01,
    e_level_02,
    e_level_max
};
```

```
CLevel m_pLevel[e_level_max];
```

Permet de rajouter facilement des entrées, et d'être garanti de ne pas en oublier sur une boucle par exemple :

```
for (int nLevel = 0 ; nLevel < e_level_max ; ++nLevel)
{
    m_pLevel[eLevel].Initialize();

    // ...
}
```