



Naviso conçoit, développe et déploie des solutions de gestion informatiques complètes destinées aux Chefs d'Entreprise et aux Cabinets d'Expertise Comptable.

Une approche purement technique touchant rapidement ses limites, Naviso intègre à ses outils technologiques une approche organisationnelle métier.

Notre objectif est de vous faire bénéficier de réelles solutions collaboratives et évolutives, afin de vous accompagner de façon durable et pérenne dans la gestion de vos activités.

Rejoignez nous sur <http://www.naviso.fr>



Une Gameboy dans WP7



7

Samuel Blanchard

Responsable développement
Blog.naviso.fr

Objectif

- Créer un émulateur Gameboy sur WP7
- Framework .NET, langage C#
- UI soignée et rapide à mettre en place (Hub Gameboy)
- La Gameboy doit être fluide

C'est parti !

- GameBoy
 - Les entrailles de la bête
 - Pause émulation (en c#)
- Implémentation WP7
 - UI Silverlight
 - Ecran XNA



Qu'est ce qu'une Gameboy ?



Ram & Rom

Mémoires et accès au registres,
cartouches de différentes tailles

Processeur

Z80 modifié, 8bits, 4MHZ

Processeur Vidéo

Ecran 160x144, 4 dégradés de gris, Scrolling,
40 Sprites, Fenêtre.

La mémoire

Tout est connecté à la mémoire !

- Contient :
 - La mémoire RAM
 - La mémoire Vidéo
 - Les registres des différents puces (Vidéo, Timer)
 - La ROM contenue dans la cartouche (et la RAM)
- Accessible par son adresse (de 0x0000 à 0xFFFF)

Plan d'adressage



La cartouche

- S'insère dans les emplacements Rom (#0 et Banks) & Ram Banks
- Possède un entête spécifique (nom, capacité)
- contient du code et des données.
- Peut posséder de la RAM, Batterie...
- ROM et RAM peuvent être échangés (MBC)
- Le code commence toujours à l'adresse 0x100

Emulation : Mémoire

- Tableau octets
- Fichier de la ROM lu dans ce tableau
- Certain accès à cette mémoire seront contrôlés
 - Pas d'écriture en ROM
 - Envoie dans les registres (Vidéo, Timer)

Emulation : Mémoire

```
public byte Lecture8Bits(int adresse)
{
    switch( adresse )
    {
        case 0xFF43 :
            // Lecture du registre de scrolling de l'écran
            // en X câblé sur le processeur vidéo
            return this.video.ScrollX;
    }

    return this.MemoireInterne[ adresse ];
}
```

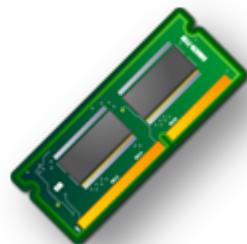
Agir grâce au CPU

- Processeur 8 bits : Z80 modifié
- 8 registres 8 bits, 2 registres 16 bits
- Ecrire et lire le contenu de la mémoire
- Exécuter du code pour manipuler ce contenu

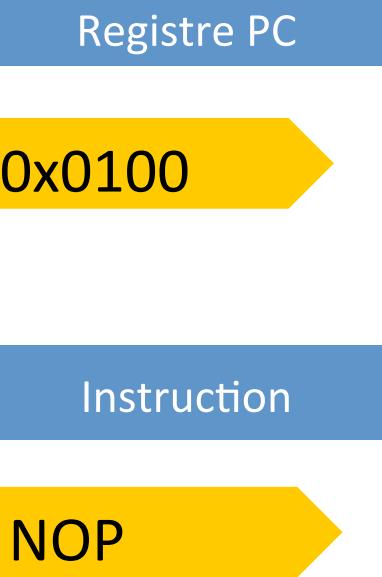
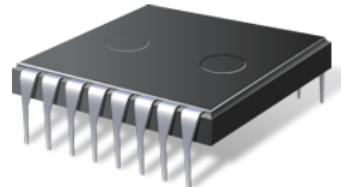
Les registres du CPU



Exécution du code



Adresse	Data
0x00FD	0x00
0x00FE	0x00
0x00FF	0x00
0x0100	0x00
0x0101	0xC3
0x0102	0xA5
0x0103	0x01



Quelques Instructions

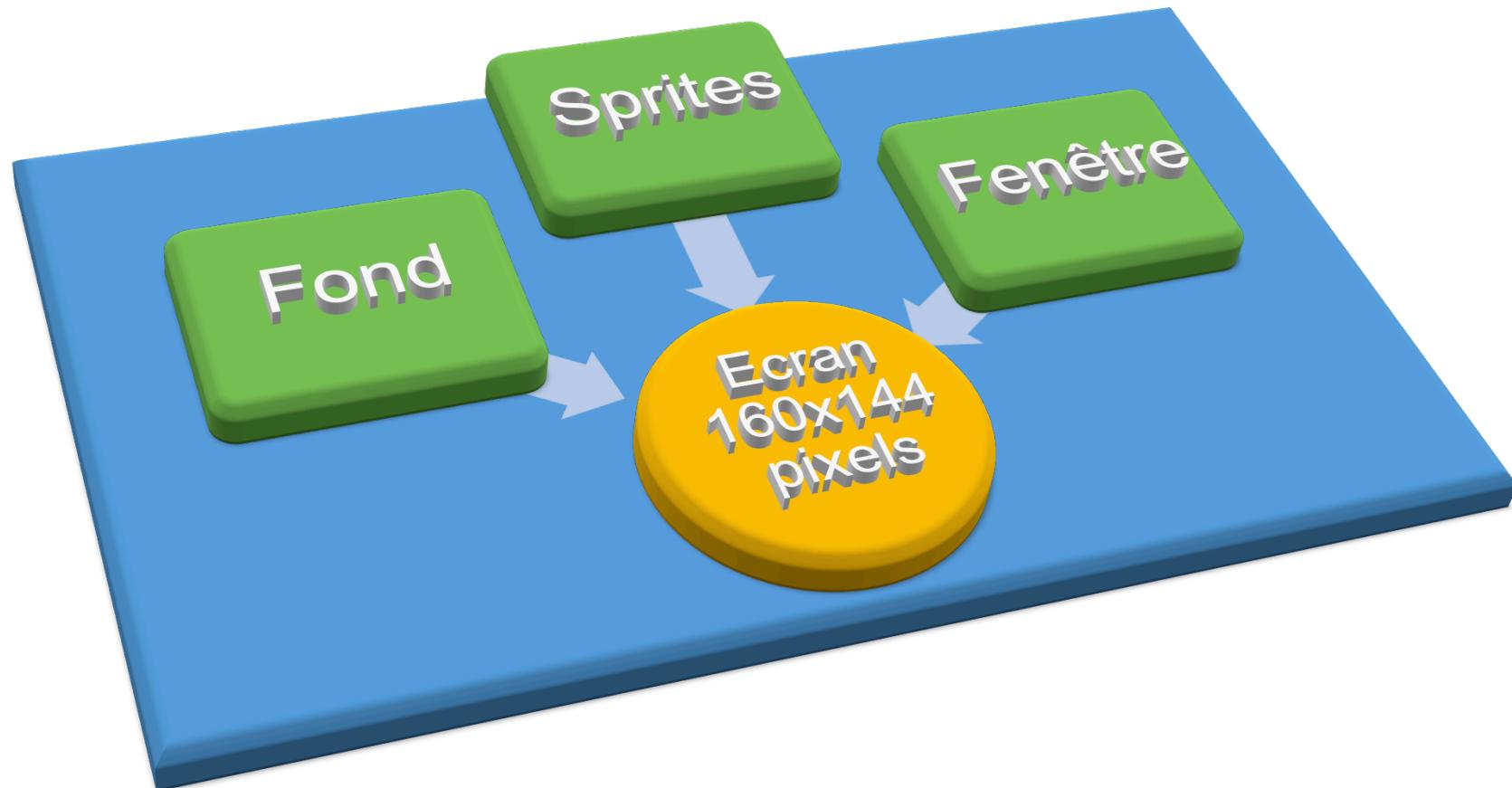
Instructions				
Nom	Paramètres	Opcodes	Cycles	
LD	AD	0x7A	4	
PUSH	HL	0xE5	16	
ADD	A [#]	0xC6	8	
JP	#	0xC3	12	

Emulation : CPU

```
byte opcode = this.Memoire.Lecture8Bits(this.PC);

Switch( opcode )
{
    // Jump
    case 0xc3 :
        this.PC = this.Memoire.Lecture16Bits(this.PC+1);
        this.Cycles += 12;
        break;
    case ...
}
```

Processeur Vidéo



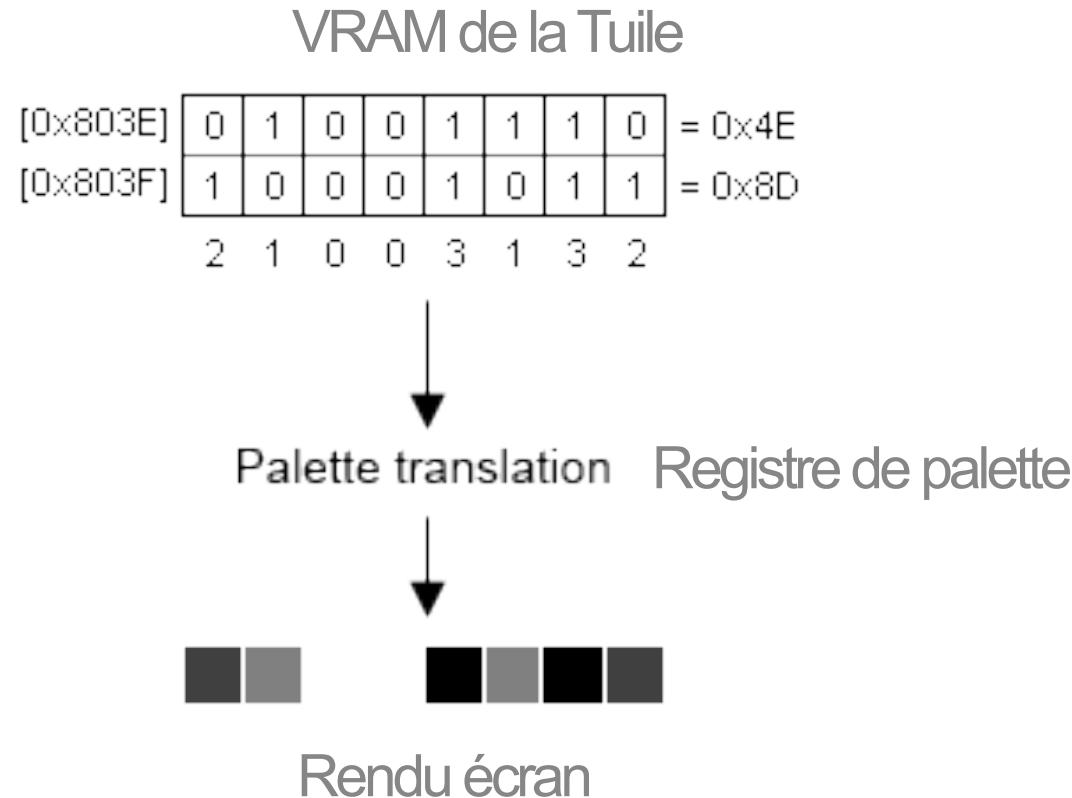
le rendu du GPU



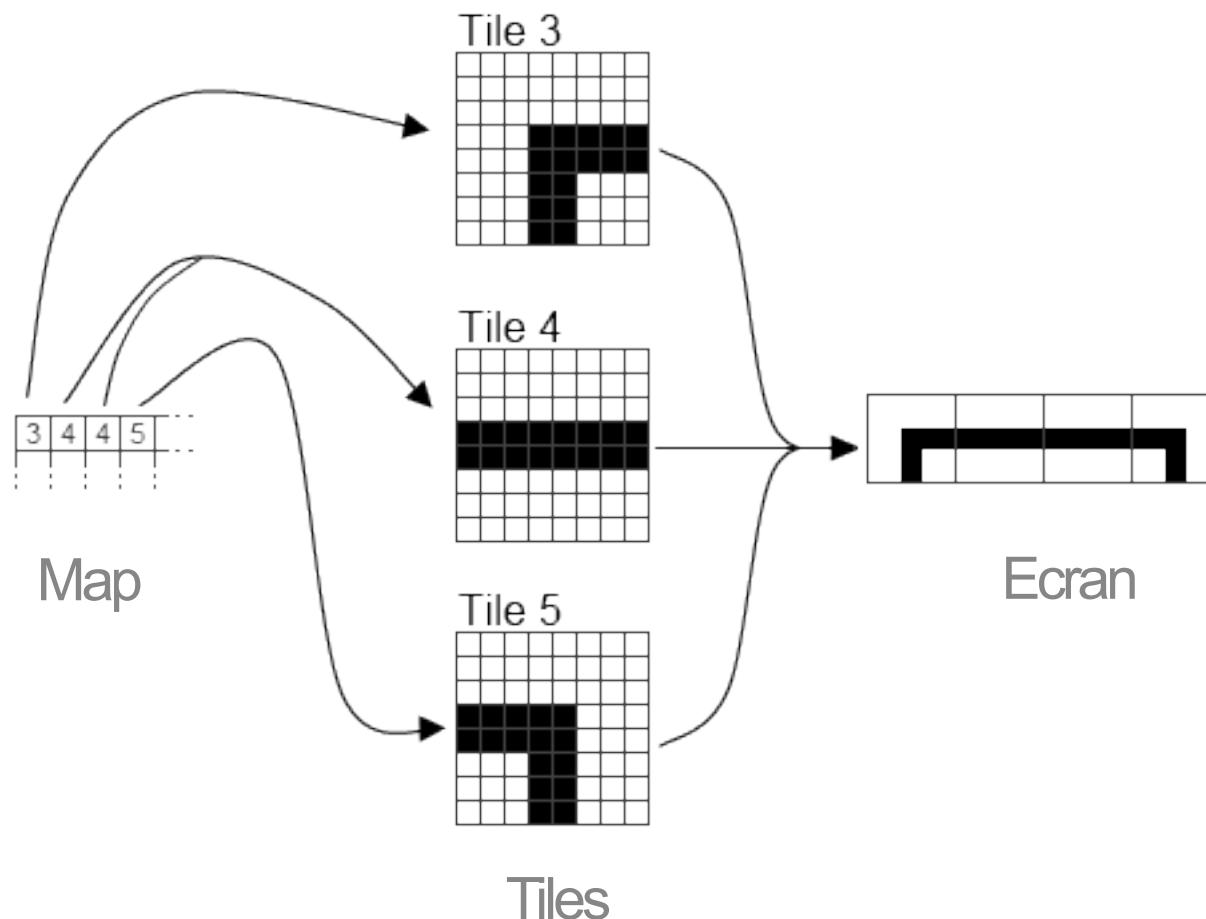
Technique des Tuiles

- Comment contrôler beaucoup de pixels avec un petit CPU et peu de mémoire ?
- Tuile ou Tile = blocs de 8x8 px dans la VRAM
- Carte ou Map = indexes des tuiles pour constituer l'image
- Sur un écran 160x144 px il y a 20x18 tuiles
 - Pixels = 23040 écriture à la VRAM
 - Tuiles = 360 écriture à la VRAM

Des pixels dans la tuile



Mécaniques des tuiles



Vidéo Ram (VRAM)



Le fond

- Fond scrollable horizontalement & verticalement
- 32x32 tiles constituent le fond (256x256 px)
- Boucle sur lui même
- Seules 20x18 tiles affichables à l'écran (160x144 px)

La fenêtre

- Fenêtre déplaçable (Barre de score)
- 20 x 18 affichables à l'écran (taille écran)

Les sprites

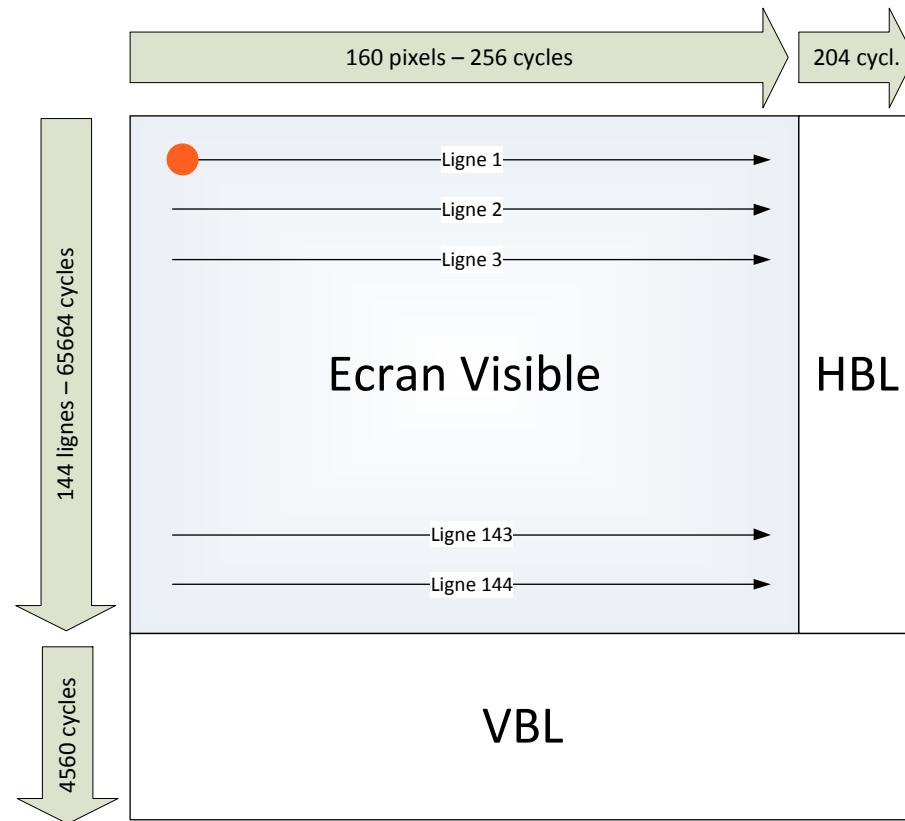
- 40 sprites simultanément, 10 sprites par ligne
- 2 modes : 1 tile ou 2 attachés verticalement
- Inversable horizontalement ou verticalement



Quelques registres vidéo

- LCDC : sélection affichage Fond, Sprites et Fenêtre et écran LCD.
- SCX/SCY : Scrolling Fond
- WXWY : Position Fenêtre
- BGP : Palette Fond et Fenêtre (4 niveaux de gris)
- OBP0/OBP1 : 2 Palettes pour les sprites (3 niveaux de gris)

L'écran LCD



Dessiner l'écran

- 160x144 pixels à représenter mêlant Fond, Fenêtre et Sprites
- HBL & VBL calculés en cycle CPU
- Rendre chaque HBL de la ligne en cours en prenant en compte les registres vidéo
- L'affichage complet à la fin de la VBL dans un tableau d'octets

Emulation : L'écran

```
protected void DrawScreenLine()
{
    int lcdc = this.memory.GetRawValue8( REGISTER_LCDC );
    if( (lcdc & VideoGB.MASK_LCDC_DISPLAY) ==
VideoGB.MASK_LCDC_DISPLAY )
    {
        this.DrawBackgroundLine( lcdc );
        this.DrawwindowLine( lcdc );
        this.DrawSpriteLine( lcdc );
    }
}
```

Emulation : Fenêtre 1

```
protected void DrawwindowLine( int lcdc )
{
    if( (lcdc & VideoGB.MASK_LCDC_WIN_DISPLAY) ==
VideoGB.MASK_LCDC_WIN_DISPLAY )
    {
        int wx = this.memory.GetRawValue8
( videoGB.REGISTER_WX );
        int wy = this.memory.GetRawValue8
( videoGB.REGISTER_WY );

        if( this.countHBL >= wy )
        {
            int addressTile;
            int addressMap;
            ...
        }
    }
}
```

Emulation : Fenêtre 2

```
// récupération du jeu de tuiles
if( (lcdc & videoGB.MASK_LCDC_TILE) ==
videoGB.MASK_LCDC_TILE )
    addressTile = 0x8000;
else
    addressTile = 0x8800;
// récupération de la map
if( (lcdc & videoGB.MASK_LCDC_WIN_MAP) ==
videoGB.MASK_LCDC_WIN_MAP )
    addressMap = 0x9c00;
else
    addressMap = 0x9800;

int lineY = (this.countHBL - wy) & 0xff;
addressMap += (lineY / 8) * 32;
...
```

Emulation : Fenêtre 3

```
int tileNumber;
int tileNumberSigned = 0;
int lineY = (this.countHBL - wy) & 0xff;
addressMap += (lineY/8) * 32;
int tileNumber;
int tileNumberSigned = 0;
if( addressstile != 0x8000 )
    tileNumberSigned = 128;

for( int x = 0; x < 20; x++ ){
    tileNumber =this.memory.GetRawValue8(addressMap+x);
    tileNumber = (tileNumber+tileNumbersigned)&0xff;
    this.DrawTileLine(addressstile+
(tileNumber*16),lineY , x*8,this.countHBL);
}}}}
```

La Gameboy

- Initialise tout ses composants (mémoire / vidéo / cpu) + chargement de la rom + de l'état
- Lance une frame complète
- rend disponible l'écran sous forme de tableau
- Arrêt (nettoyage et sauvegarde de l'état)

Emulation : une frame

```
public byte[] RunOneFrame()
{
    do
    {
        int cycle = this.processor.Run();

        this.timer.Run(cycle);
        this.video.Run(cycle);
    }
    while (this.video.HaveNewFrame == false);

    return this.video.Screen;
}
```

TODO

- Indispensable :
 - MBC : Contrôleur de banque mémoire
 - Interruption
 - JoyPad
 - DMA
- Optionnel :
 - Timer
 - Son
 - Port Série

Du côté de WP7 !



Système d'exploitation

Crée à partir d'une feuille blanche.
Incompatible Windows Mobile.

Interface utilisateur

UI novatrice appelée Metro : concept similaire dans Zune, Windows 8.
Gestion de Hub, Icône variant selon utilisation,...

Développement

Tout en .NET que ce soit pour Silverlight ou XNA. Langage C# et bientôt VB.NET.
Outils de développement performants

OS : Nodo et Mango

- Ce sont deux versions de l'OS de WP7 :
 - Nodo est la version courante (Copier-coller)
 - Mango la version qui arrivera à l'automne 2011
- Mango propose des nouvelles APIs coté Silverlight qui vont nous intéresser

Présentation Metro WP7

- UI simple et intuitive
- Ecran démarrer et vignette dynamique
- Hub : Regroupement d'informations
 - Contacts
 - Photos + caméra
 - Musiques + vidéos

Présentation Metro WP7



Les outils

- Visual Studio 2010 : Editeur de code et d'UI (pour Silverlight)
- La suite expression :
 - Expression Design : Dessin vectoriel avec export XAML
 - Blend 4 : Editeur code, UI et animation.

Silverlight ou XNA ?

- WP7 propose deux frameworks de développement :
 - XNA pour les jeux
 - Silverlight pour les applications standards
- XNA est rapide pour les accès pixels
- Silverlight possède des contrôles de haut niveaux

Le meilleur des 2 mondes !

- Mixer les qualités des deux
- UI Silverlight et appel d'un écran XNA !
- Selon la version de l'OS :
 - Nodo : Silverlight et appel écran XNA via un trick.
 - Mango : possède une classe dédiée dans Silverlight !

Partage des tâches

- XNA :
 - Affichage de l'écran de la Gameboy
- Silverlight :
 - Interface de sélection de jeu
 - Téléchargement du jeu
 - Lancement / Fonctionnement de la Gameboy

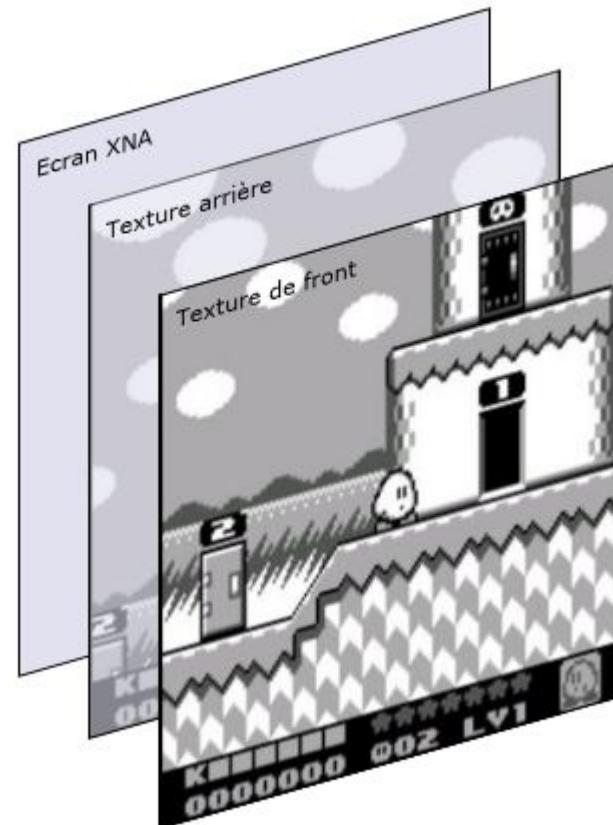
XNA : Affichage écran

- Dans l'API Mango
- Classe GameTimer
 - 2 évènements générer à intervalle régulier (30/s)
 - Update : Créer l'écran
 - Draw : Afficher l'écran
- Pas de réel mélange XNA / Silverlight

XNA : mécanisme écran

- Dessiner notre écran Gameboy dans une texture.
- modification d'une texture affichée impossible
- 2 Textures :
 - Front : destinée à l'affichage
 - Back : destinée à la modification
- Inverser les textures à chaque VBL

XNA : mécanisme écran



XNA : Affichage écran

```
public override void OnDraw(SpriteBatch batch)
{
    // on remplit la texture des pixels de la GB
    this.backScreen.SetData<T>(this.Pixels);

    // Permutation des texture
    Texture2D temp = this.backScreen;
    this.backScreen = this.frontScreen;
    this.frontScreen = temp;

    // Affichage de la texture front
    batch.Draw(this.frontScreen, this.Destination,
    this.Color);
}
```

SL : Hub Gameboy



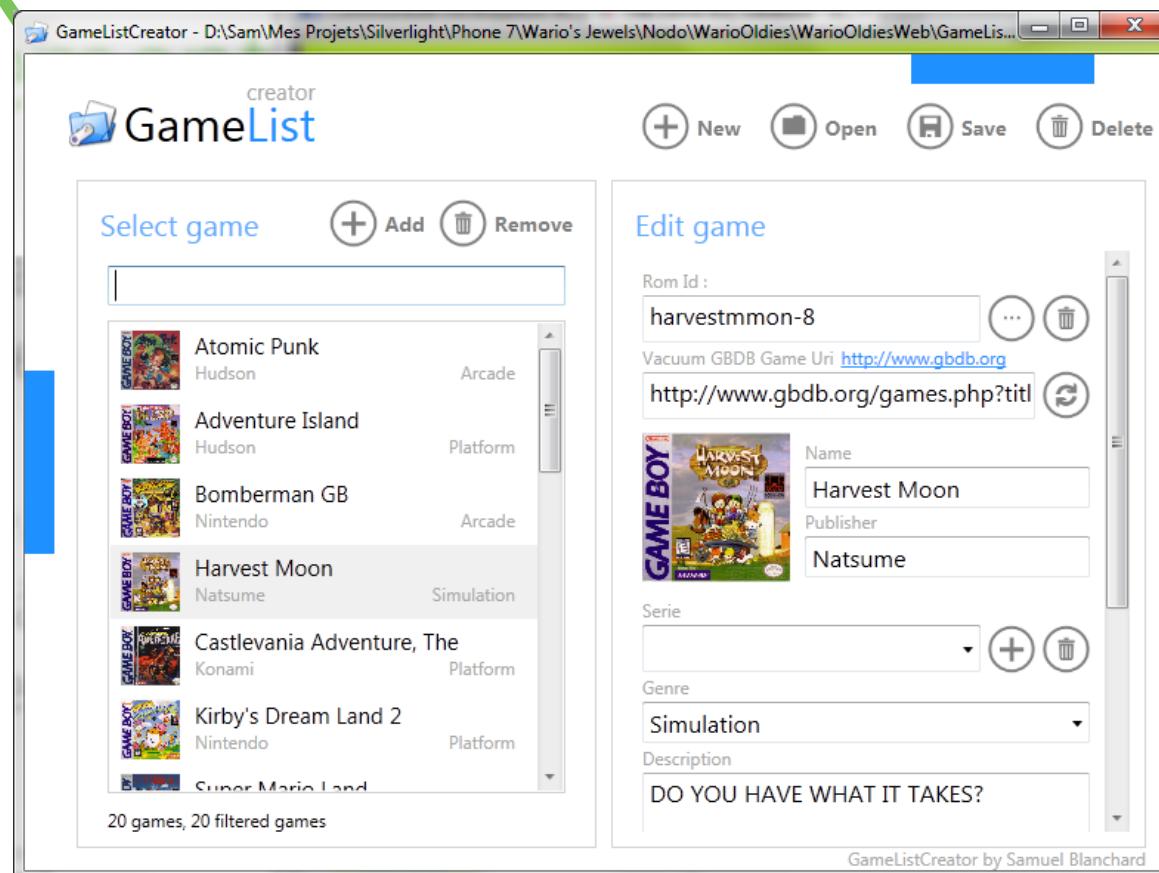


Demo – Panorama

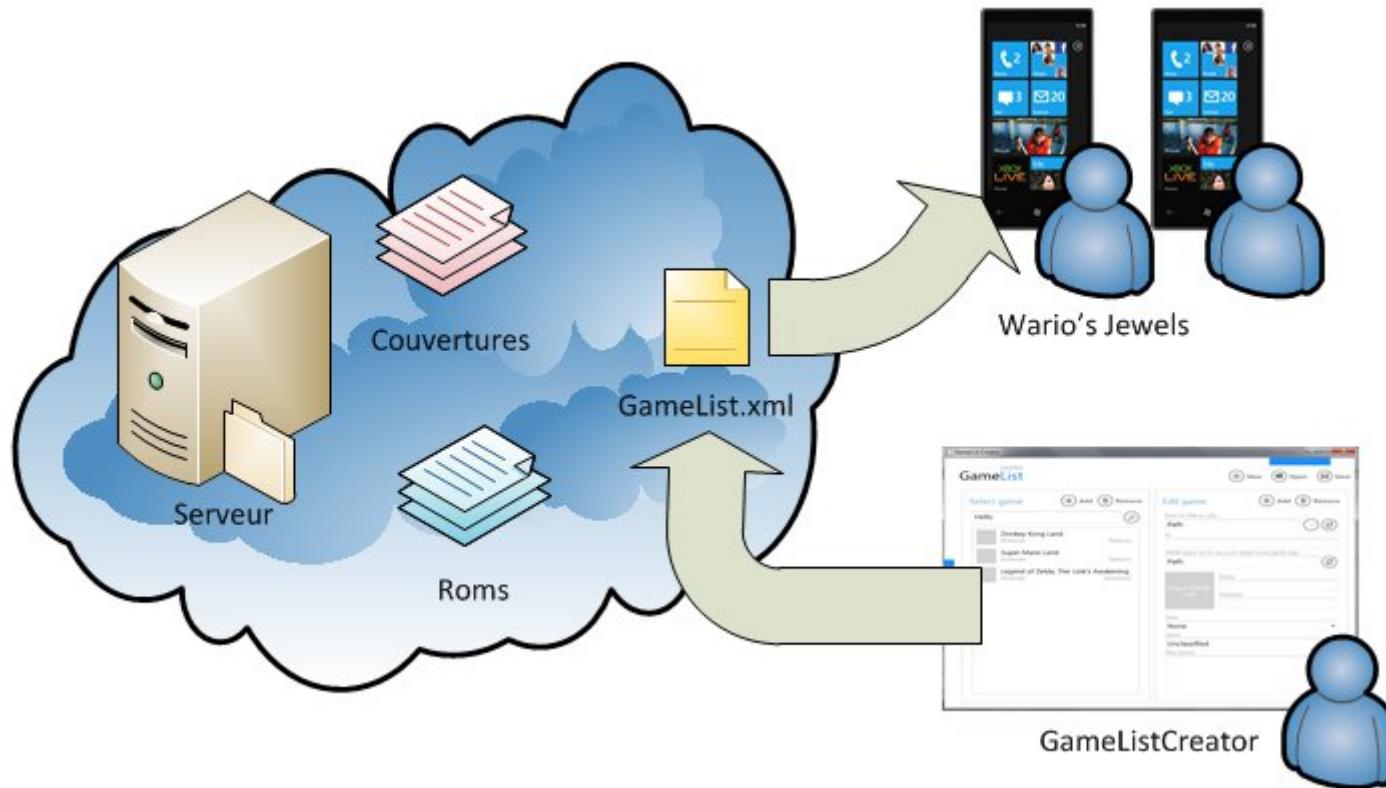
Téléchargement

- Crédit d'une liste de jeux (GameListCreator)
 - dépôt sur un serveur
- Récupération de la liste des jeux par WJ7
 - Internet
 - Cache (IsolatedStorage)
- Téléchargement du jeux
 - Internet
 - Cache (IsolatedStorage)

Création de la liste de jeux



Téléchargement





Demo – WJ7

Wario's Jewels 7

- Des infos sur WJ7
 - Sur mon Blog : blog.naviso.fr
 - Sur twitter : [samoteph](https://twitter.com/samoteph)
- WJ7 Sortira prochainement en version Nodo puis à l'automne en version Mango
- GameListCreator
 - Codeplex : gamelistcreator.codeplex.com

Documentation

- Site de vulgarisation sur la Gameboy par Imran Nazar
 - <http://imrannazar.com/GameBoy-Emulation-in-JavaScript:-The-CPU>
- Documentation Gameboy
 - <http://www.scribd.com/doc/39999184/GameBoy-Programming-Manual>
- Documentation WP7
 - <http://msdn.microsoft.com/fr-fr/windowsphone>

Remerciements

- Le Club Sharepoint & .NET Ouest et plus particulièrement Guillaume Collic
- Michael Hyot (Naviso) pour son implication dans le communautaire et ses encouragements
- Alessandra Sada pour son soutien

Please read (hidden slide)

- This template is designed for use with Office PowerPoint 2007 and 2010. The charts and graphics can be edited with PowerPoint 2007 and 2010, but not with PowerPoint 2003.
- This template uses Microsoft's corporate font, Segoe Light.
- Segoe is not a standard font included with Windows, so it has been embedded in this template.
- For Live Meeting situation, please use Segoe Regular

More resources:

- Learn how to use this PowerPoint template by referring to the **Windows Phone 7 Guidelines**.
- Find more pre-designed layouts by referring to the **Windows Phone 7 Additional Layouts**.

Q & R



Naviso conçoit, développe et déploie des solutions de gestion informatiques complètes destinées aux Chefs d'Entreprise et aux Cabinets d'Expertise Comptable.

Une approche purement technique touchant rapidement ses limites, Naviso intègre à ses outils technologiques une approche organisationnelle métier.

Notre objectif est de vous faire bénéficier de réelles solutions collaboratives et évolutives, afin de vous accompagner de façon durable et pérenne dans la gestion de vos activités.

Rejoignez nous sur <http://www.naviso.fr>