Bien démarrer son projet

Valentin 'toogy' Iovene
Alexis 'Horgix' Chotard
Théophile 'yroeht' Ranquet

La conférence

1. Organiser son projet et maintenir son code

La conférence

1. Organiser son projet et maintenir son code
2. Git et le versioning

La conférence

1. Organiser son projet et maintenir son code
2. Git et le versioning
3. LaTeX

La conférence

1. Organiser son projet et maintenir son code
2. Git et le versioning
3. LaTeX
4. DirectX et C#

La conférence

1. Organiser son projet et maintenir son code
2. Git et le versioning
3. LaTeX
4. DirectX et C#
5. Introduction à XNA

La conférence

1. Organiser son projet et maintenir son code
2. Git et le versioning
3. LaTeX
4. DirectX et C#
5. Introduction à XNA
6. TP Git & XNA par équipe

1. Organiser son projet et maintenir son code

Organiser son **projet**

Réfléchir avant d'agir (de coder)

Avoir les idées claires, savoir où on va

Avoir les idées claires, savoir où on va

- Brainstorming : trouver ce qu'on **veut** faire

Avoir les idées claires, savoir où on va

- Brainstorming : trouver ce qu'on **veut** faire
- Fixer un but *concret* à atteindre (le produit)

Avoir les idées claires, savoir où on va

- Brainstorming : trouver ce qu'on **veut** faire
- Fixer un but *concret* à atteindre (le produit)
- Représenter son projet visuellement

Découper son projet en fonctionnalités

Découper son projet en fonctionnalités

- Menu

Découper son projet en fonctionnalités

- Menu
- Minimap

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments
  - Combats

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments
  - Combats
  - Déplacements

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments
  - Combats
  - Déplacements
- Système de ressources

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments
  - Combats
  - Déplacements
- Système de ressources
- Bâtiments

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments
  - Combats
  - Déplacements
- Système de ressources
- Bâtiments
  - File d'attente de création d'unités

Découper son projet en fonctionnalités

- Menu
- Minimap
- Map
- Unités
  - Ouvrier
    - Récolte
    - Construction de bâtiments
  - Combats
  - Déplacements
- Système de ressources
- Bâtiments
  - File d'attente de création d'unités
  - Point de ralliement

Priorisation (valeur ajoutée + dépendance)

- Map
- Unités
- Unités :Déplacements
- Système de ressources
- Bâtiments
- Unités :Ouvrier :Récolte
- Unités :Ouvrier :Construction de bâtiments
- Unités :Combats
- Bâtiments :File d'attente de création d'unités
- Bâtiments :Point de ralliement
- Minimap
- Menu


KNIFE THE BABY

Maintenir son **{code}**

Field (champ)

```
public class Truc
{
    public int Attribut;
}
```

Field (champ)

```
public class Truc
{
    public int Attribut;
}
```

```
public class ...
{
    public void MaFonction(...)
    {
        Truc machin = new Truc();

        Console.Write(Truc.Attribut);
    }
}
```

Field (champ)

```
public class Truc
{
    public int Attribut;
}
```

```
public class ...
{
    public void MaFonction(...)
    {
        Truc machin = new Truc();

        Console.Write(Truc.Attribut);
    }
}
```

Property (propriété)

Snippet VS : propfull

```
public class Truc
{
    // backing field
    private int _attribut;

    public int Attribut // propriété
    {
        get { return _attribut; }
        set { _attribut = value; }
    }
}
```

Validation

```
class Thermostat
{
    private int _temperature; // backing field

    public int Temperature // propriété
    {
        get { return _temperature }
        set
        {
            if(value >= 50)
                _temperature = 50;
            else
                _temperature = value;
        }
    }
}
```

Auto-propriété

Snippet VS : prop

```
public class Objet
{
    public int Attribut { get; set; };
}
```

Auto-propriété

Snippet VS : prop

```
public class Objet
{
    public int Attribut { get; set; };
}
```

Accès privé sur le *set*

Snippet VS : propg

```
public class Objet
{
    public int Attribut { get; private set; };
}
```

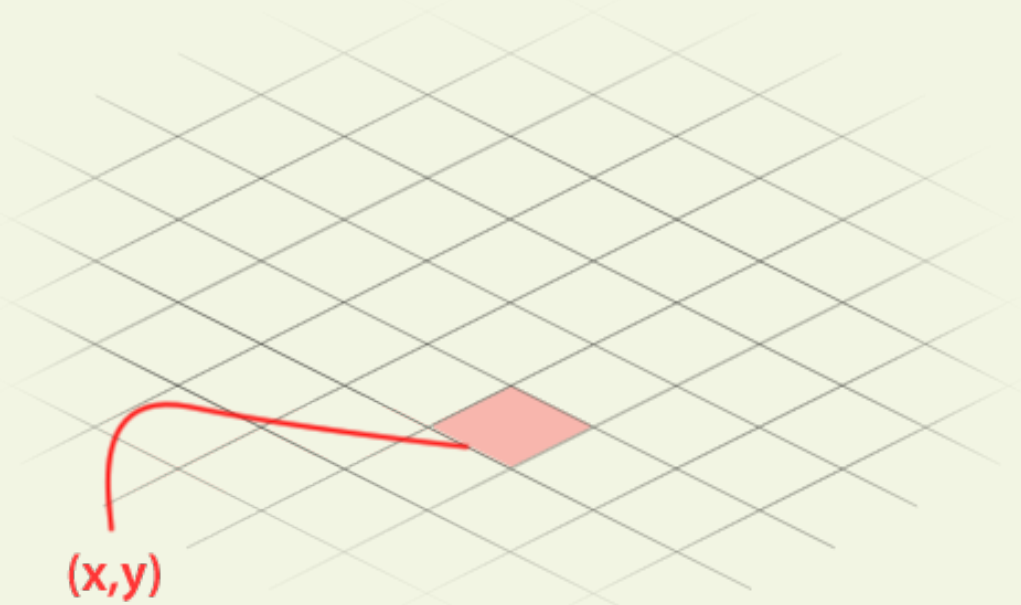
Interface

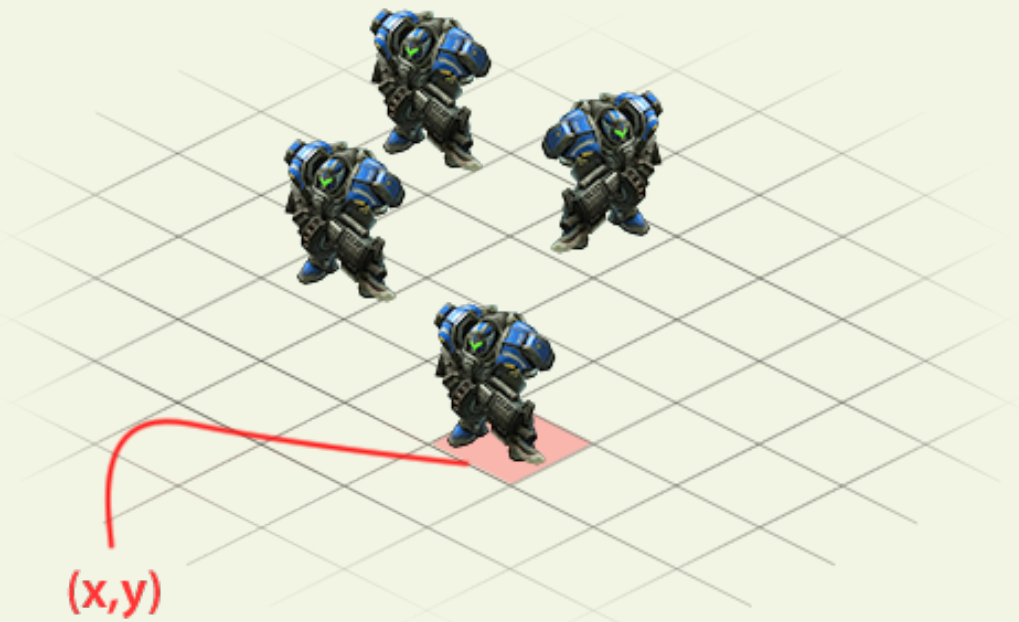
```
interface IBicycle
{
     string BrandName { get; set; }

     void ChangeSpeed(int newValue);

     void Brake();
}
```

Interface

```
interface IBicycle
{
     string BrandName { get; set; }

     void ChangeSpeed(int newValue);

     void Brake();
}
```

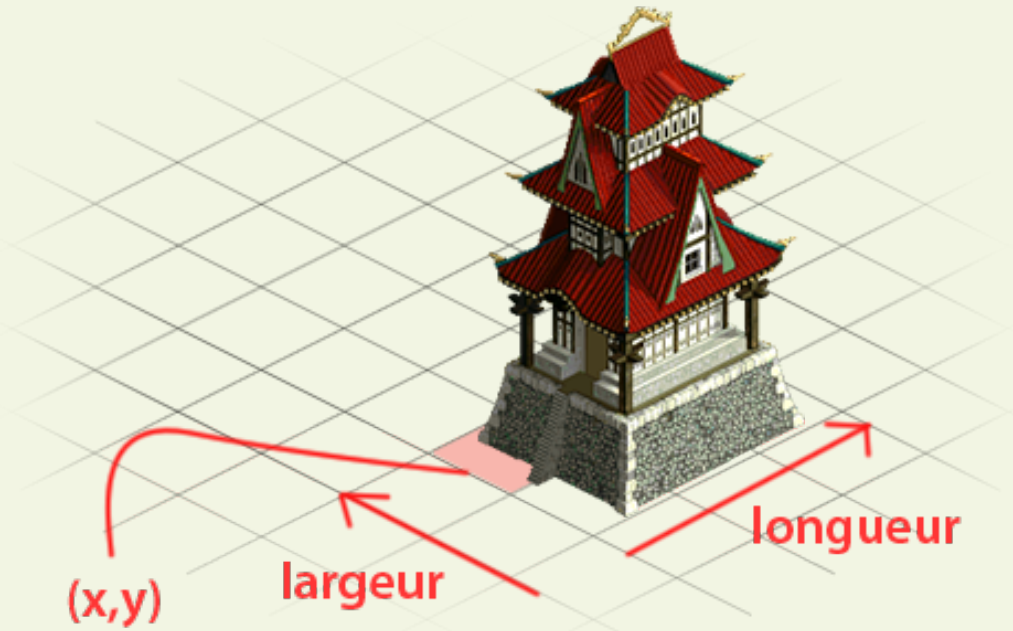
```
class BlueBicycle : IBicycle
{
    private string _brandName;

    public string BrandName
    {
        get { return _brandName }
        set { _brandName = lowerCase(value); }
    }
}
```


(x,y)


(x,y)

Modélisation d'une unité

```
class Unit
{
    public Tuple<int,int> Position { get; set; }

    public int HealthPoints { get; set; }
    private int _maxHealthPoints;

    public int Speed { get; private set; }

    public int Dps { get; private set; }

    public void Move(Tuple<int,int> destination)
    {
        // Bouger
    }

    public void Die()
    {
        // Mourir
    }
}
```


(x,y)
largeur
longueur

Modélisation d'une unité et d'un bâtiment

```
class Unit
{
    public Tuple<int,int> Position { get; set; }

    public int HealthPoints { get; set; }
    private int _maxHealthPoints;

    public int Speed { get; private set; }

    public int Dps { get; private set; }

    public void Move(Tuple<int,int> destination)
    {
        // Bouger
    }

    public void Die()
    {
        // Mourir
    }
}
```

```
class Building
{
    public Tuple<int,int> Position { get; private set; }
    public Tuple<int,int> Size { get; private set; }

    public List<Unit> Units { get; set; }

    public int HealthPoints { get; set; }
    private int _maxHealthPoints;

    public void Die()
    {
        // Mourir
    }
}
```

```
abstract class GameItem
{
    public Tuple<int,int> Position
        { get; private set; }

    public int HealthPoints
        { get; set; }

    private int _maxHealthPoints;

    public abstract void Die();
}
```

```
abstract class GameItem
{
    public Tuple<int,int> Position
        { get; private set; }

    public int HealthPoints
        { get; set; }

    private int _maxHealthPoints;

    public abstract void Die();
}
```

```
class Building : GameItem
{
    public Tuple<int,int> Size
        { get; private set; }

    public List<Unit> Units
        { get; set; }

    public override void Die()
    {
        // Mourir
    }
}
```

```
abstract class GameItem
{
    public Tuple<int,int> Position
        { get; private set; }

    public int HealthPoints
        { get; set; }

    private int _maxHealthPoints;

    public abstract void Die();
}
```

```
class Building : GameItem
{
    public Tuple<int,int> Size
        { get; private set; }

    public List<Unit> Units
        { get; set; }

    public override void Die()
    {
        // Mourir
    }
}
```

```
class Unit : GameItem
{
    public int Speed
        { get; private set; }

    public int Dps
        { get; private set; }

    public void Move(
        Tuple<int,int> destination)
    {
        // Bouger
    }

    public override void Die()
    {
        // Mourir
    }
}
```

```
abstract class GameItem
{
    public Tuple<int,int> Position
        { get; private set; }

    public int HealthPoints
        { get; set; }

    private int _maxHealthPoints;

    public abstract void Die();
}
```

```
class Building : GameItem
{
    public Tuple<int,int> Size
        { get; private set; }

    public List<Unit> Units
        { get; set; }

    public override void Die()
    {
        // Mourir
    }
}
```

```
class Unit : GameItem
{
    public int Speed
        { get; private set; }

    public int Dps
        { get; private set; }

    public void Move(
        Tuple<int,int> destination)
    {
        // Bouger
    }

    public override void Die()
    {
        // Mourir
    }
}
```

Refactorisation !

```
class ...
{
    public void Fonction(...)
    {
        List<GameItem> gameItems = new List<GameItem>();

        gameItems.Add(new Unit());
        gameItems.Add(new Building());

        ...
    }
}
```

Virtual methods

```
abstract class Truc
{
    public virtual void Fonction(int parameter)
    {
        // Faire quelque chose
    }
}
```

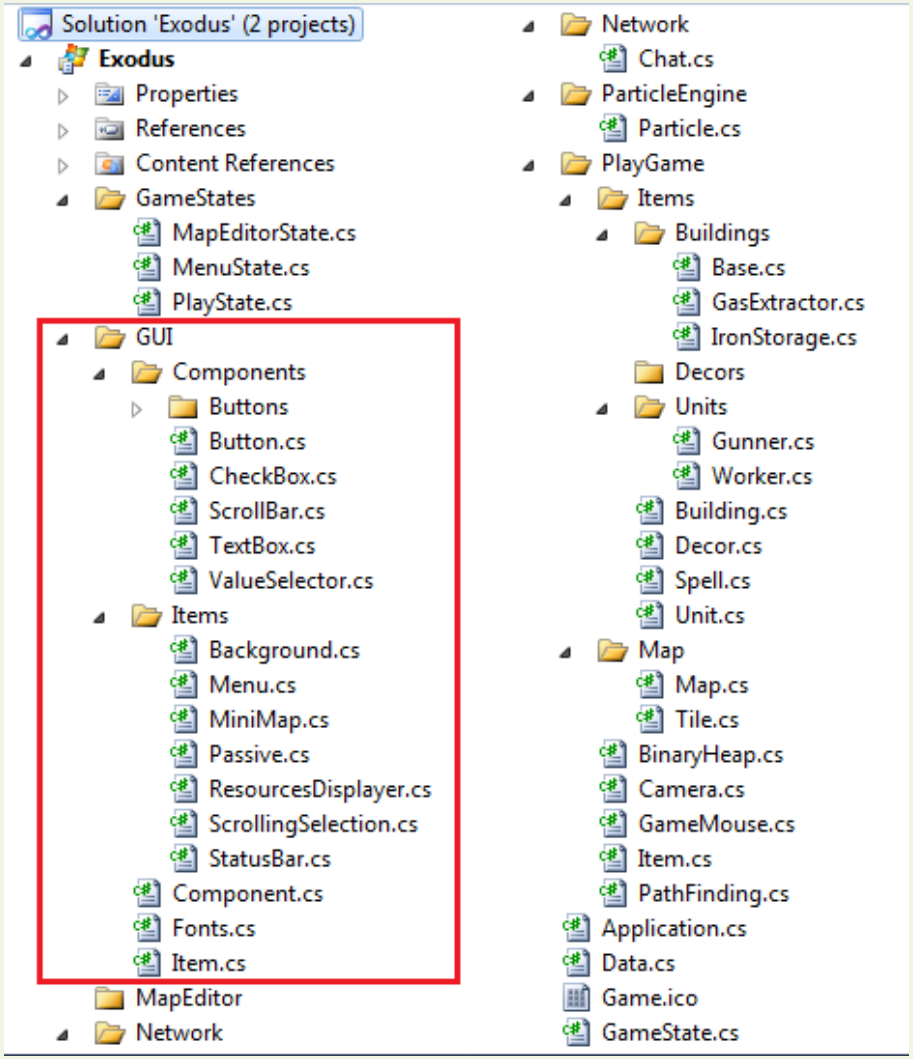
Virtual methods

```
abstract class Truc
{
    public virtual void Fonction(int parameter)
    {
        // Faire quelque chose
    }
}
```

```
class Machin : Truc
{
    public override void Fonction(int parameter)
    {
        // Faire quelque chose d'autre

        base.Fonction(parameter);
    }
}
```

```
class Machin2 : Truc
{
}
```

Architecture du code source



LOADING

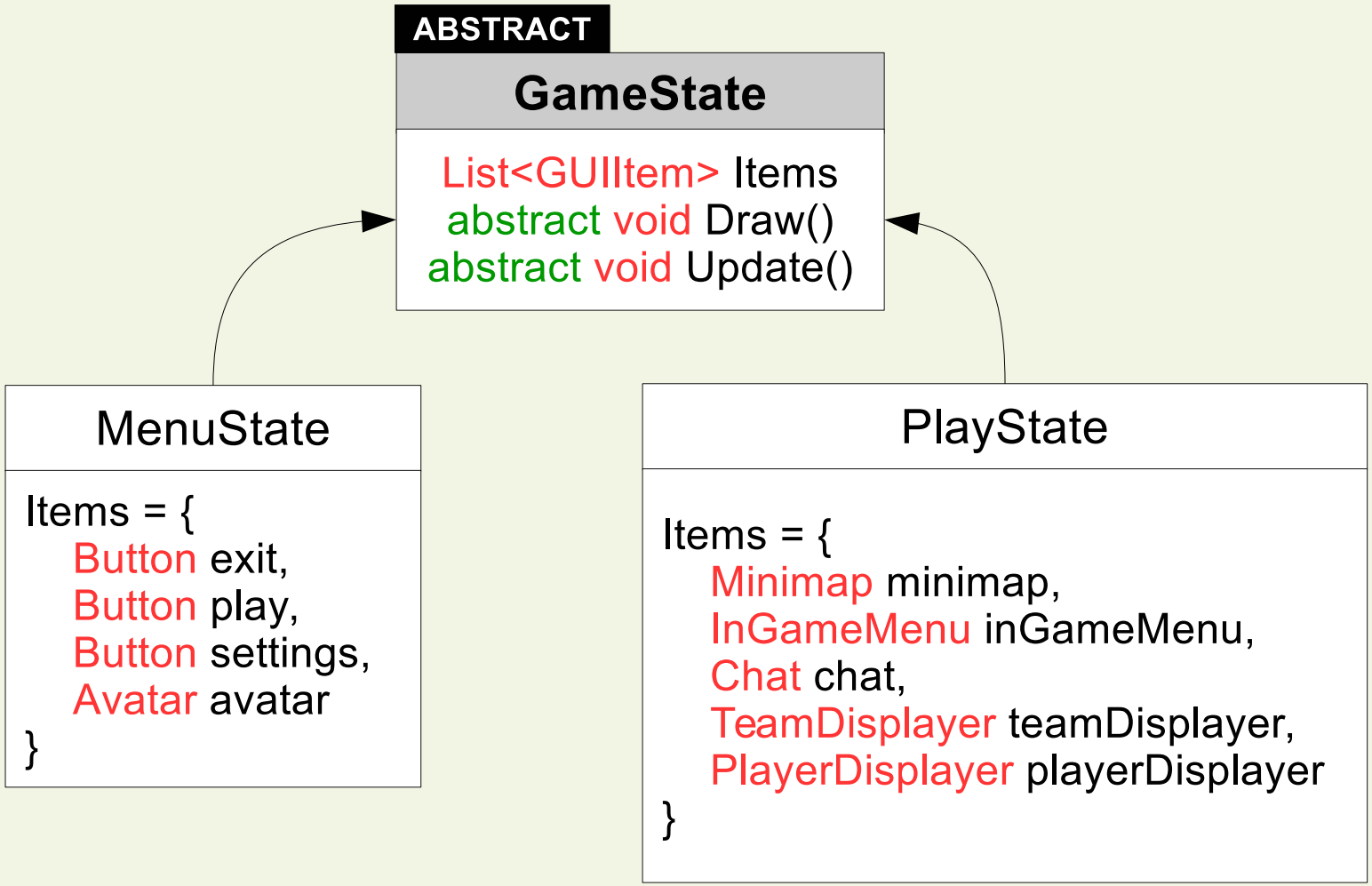
CREATE A GAME

PLAYER	MAP	CREATED
toogy	Escadron du Valdouoeuor	12:21 AM
crapeur	qxqx de l'Obsi	12:21 AM
ygoot	pupupu du Valdor	17:26 AM
Atchoumba	Rouloootheu	11:21 PM
gpth	pupu du Valdor	12:21 AM
jjjk	opupuu du oeuou	13:21 AM

JOIN THIS GAME


667 / 800


!
1
2
3
4
5
6
08:12


ABSTRACT
GameState
List<GUIItem> Items
abstract void Draw()
abstract void Update()
MenuState
Items = {
Button exit,
Button play,
Button settings,
Avatar avatar
}
PlayState
Items = {
Minimap minimap,
InGameMenu inGameMenu,
Chat chat,
TeamDisplayer teamDisplayer,
PlayerDisplayer playerDisplayer
}

2. **Git** et le versioning

Why use version control

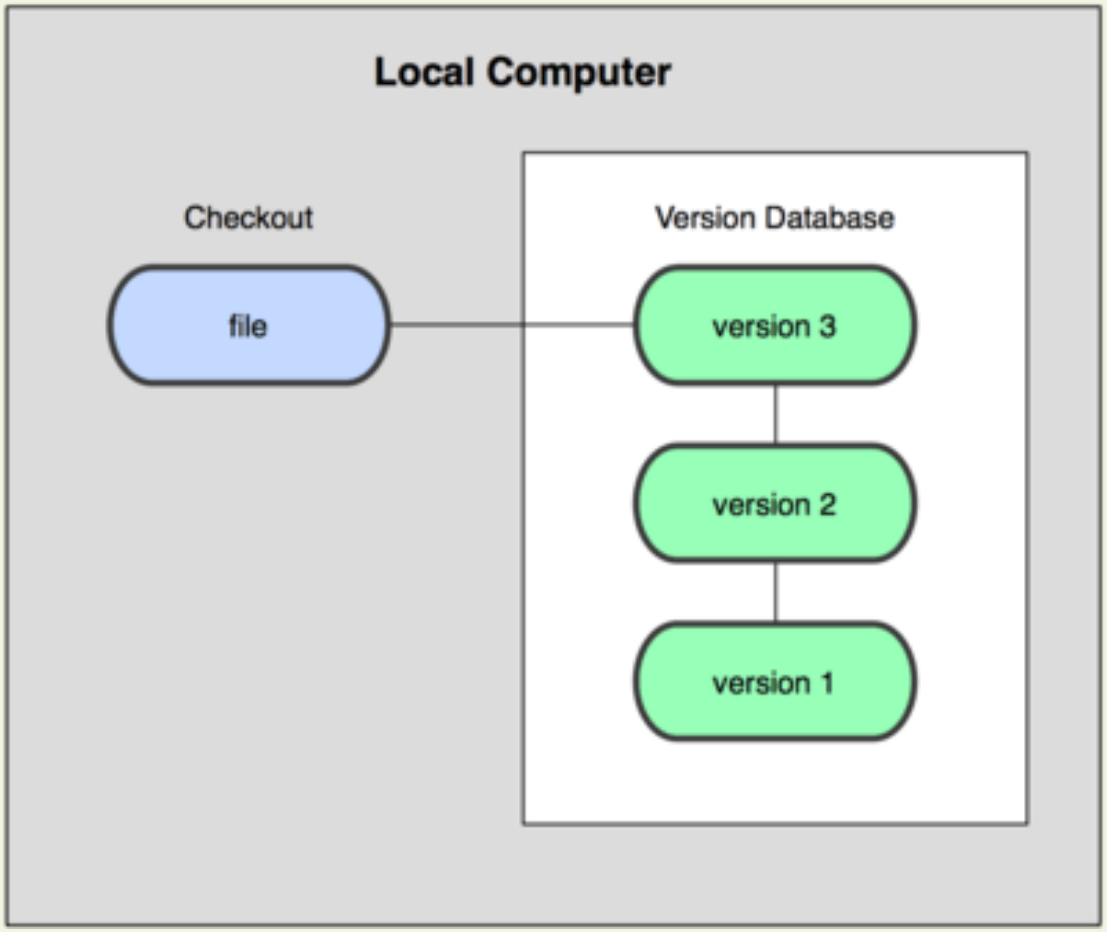
- Manage your source code (SCM)
- Keep a trace of every modification
- Come back to any previous state
- Collaborate

How does it work

cpold

```
horgix@avalon  /cpold$ ls
file file.OK file.old.old
file.2013-08-12 file.old file.old.test
file.2013-11-15.ok file.OLD file.horgix
file.to-check file.backup file.old.not-working
horgix@avalon  /cpold$
```

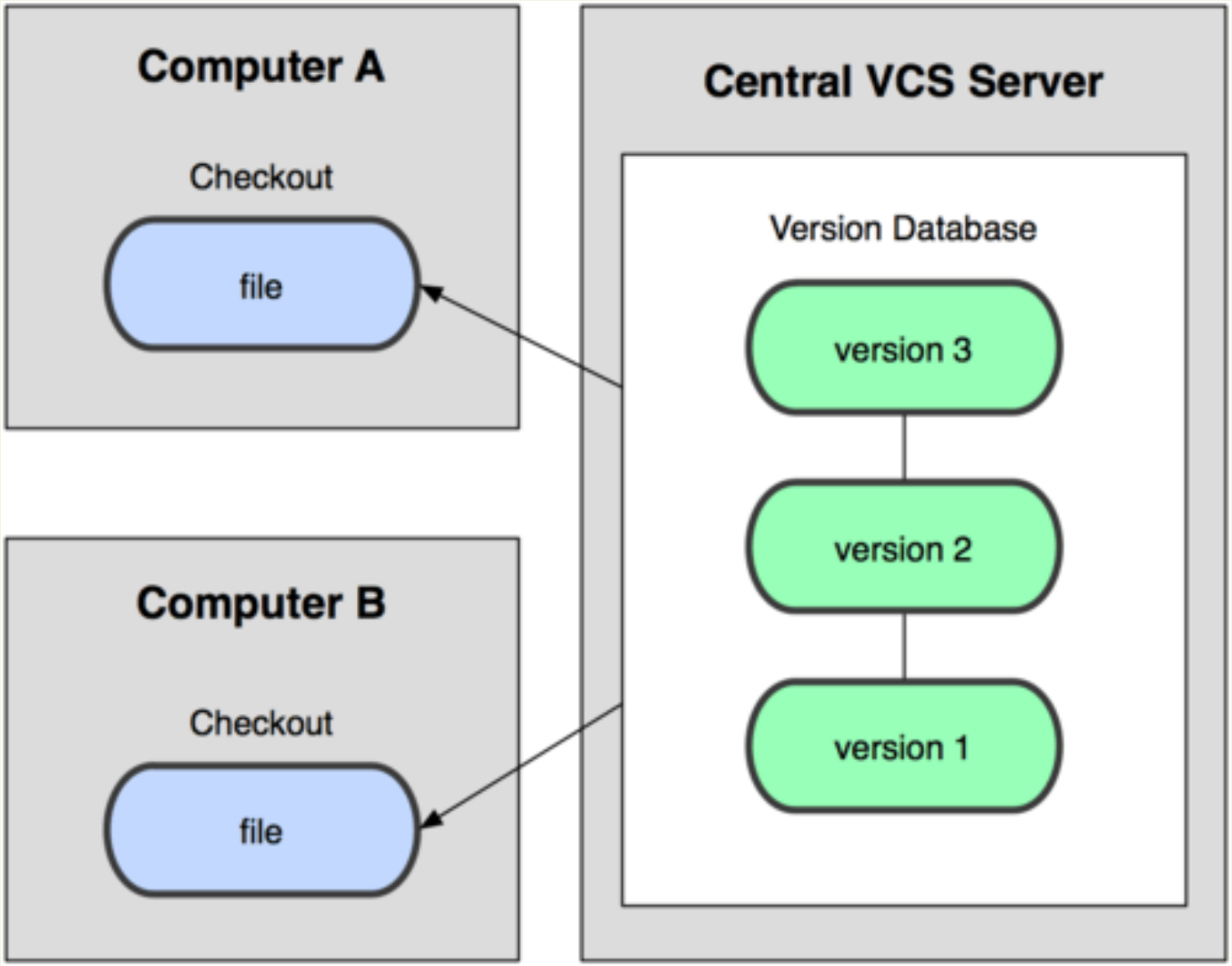
Local


Local Computer
Checkout
file
Version Database
version 3
version 2
version 1

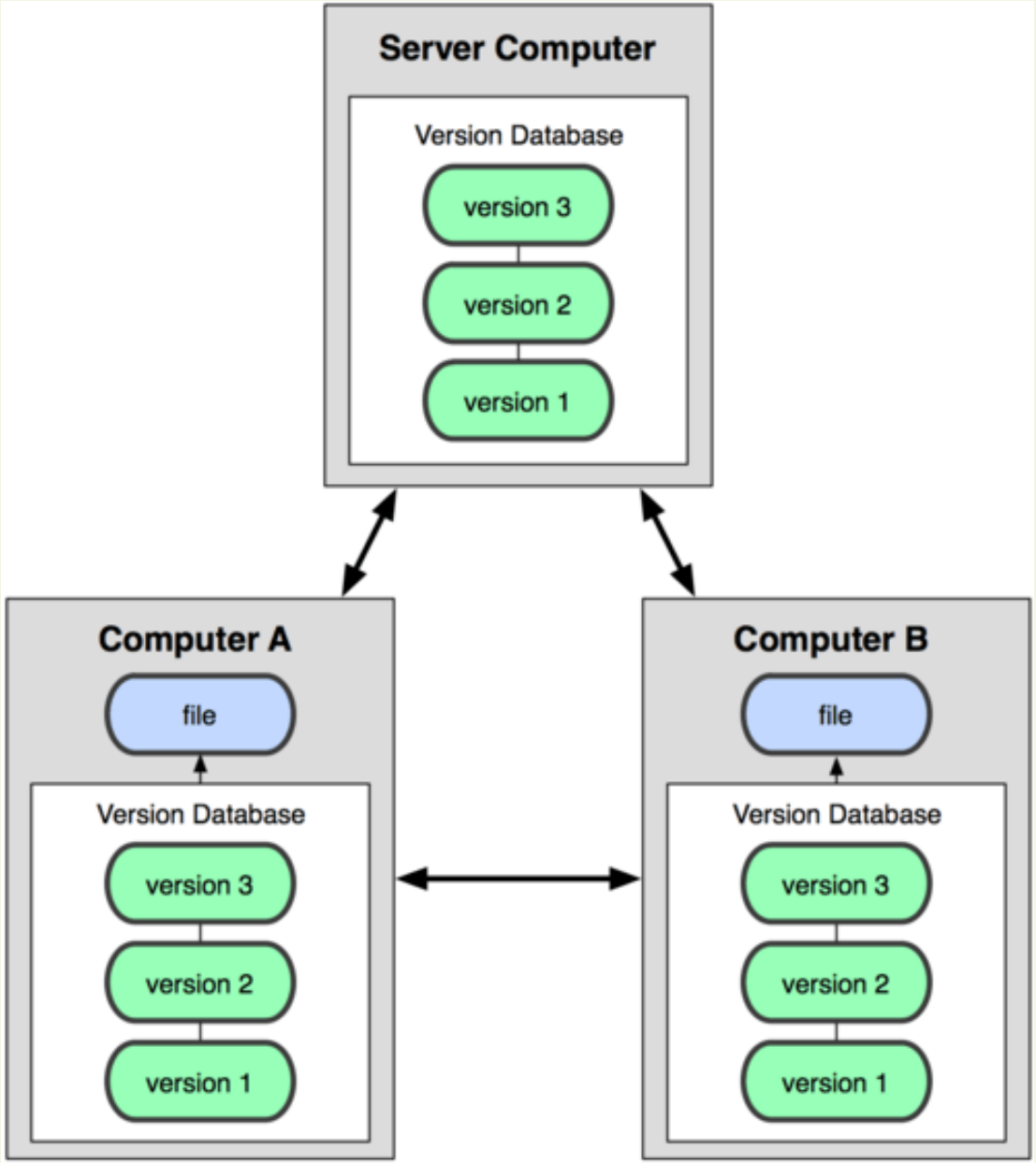
Vocabulary

- commit
- push
- pull
- revert
- merge

Centralized



Distributed



Some Version Control Systems

- Git
- Subversion (SVN)
- Mercurial
- GNU Bazaar
- CVS
- Perforce

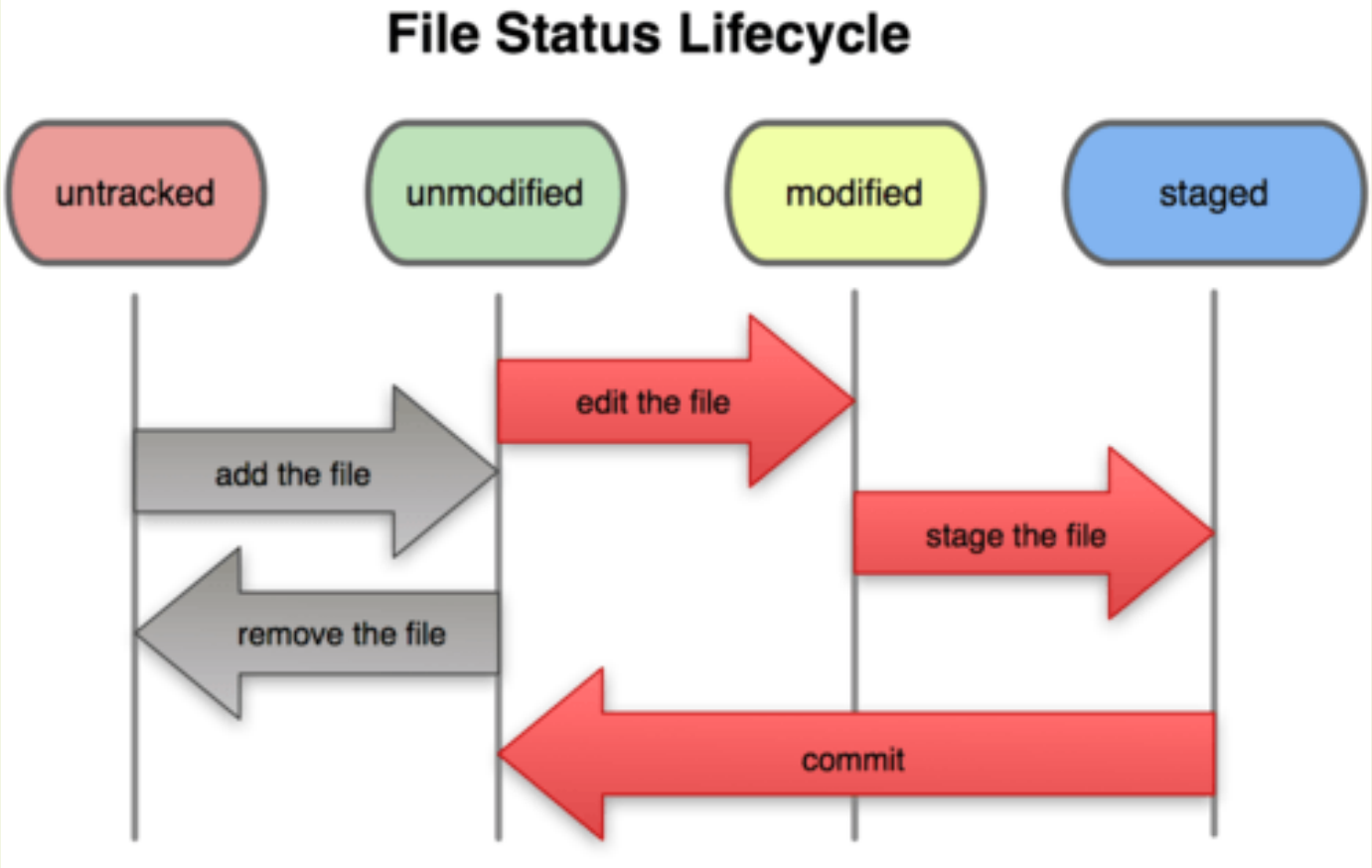
Git

"Git, c'est le système de rendu d'Epita"

- Google (https ://github.com/google)
- Facebook (https ://github.com/facebook)
- Microsoft (http ://aspnetwebstack.codeplex.com/)
- Twitter (https ://github.com/twitter)
- Perl (http ://perl5.git.perl.org/perl.git)
- Android (https ://android-review.googlesource.com)
- Git (https ://git.kernel.org/cgit/git/git.git/)
- ...

G(ot) it ?

Theory


File Status Lifecycle
untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

Practice

- clone
- add
- commit
- push
- pull
- conflicts

Good behaviors

"Commit early, commit often"
Only commit compiling code.
Use explicit commit messages.
Don't add :

- Binaries (.exe, .o, .out, .pdf, ...)
- System files (Thumbs.db, ...)
- Useless files (git is not a file sharing system)

Where can I host my git repository

GitHub - https ://github.com/

- 5M users
- Unlimited contributors
- Web pages hosting (GitHub Pages)

Bitbucket - https ://bitbucket.org/

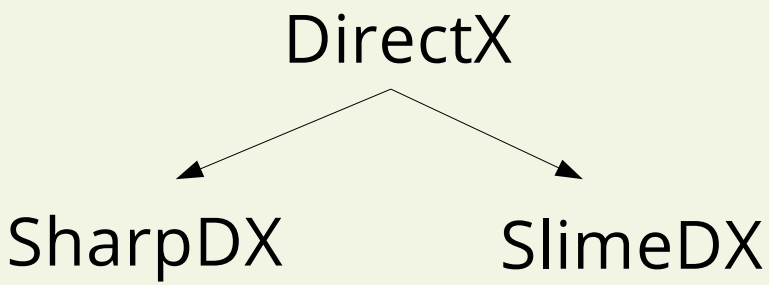
- 1M users
- Unlimited free repositories
- External authentication (Facebook, OpenID, Google)à
- Also supports SVN and Mercurial

Links

- http ://git-scm.com/
- git.acu.epita.fr

3. LaTeX

4. DirectX et C#


DirectX
SharpDX
SlimeDX

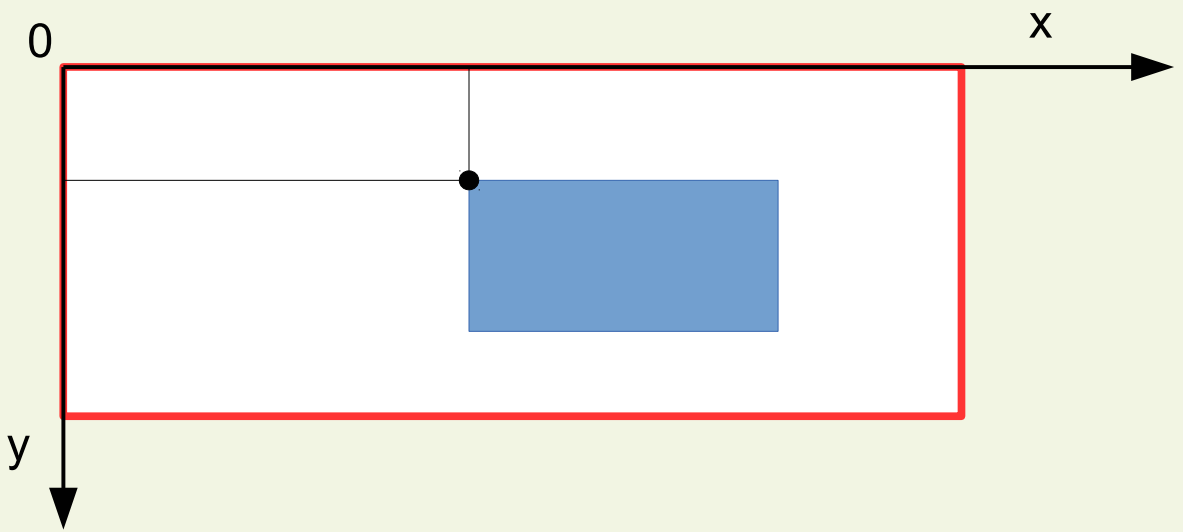
SlimeDX

+	-
Docs	Pas à jour
Tutos	Abandonné

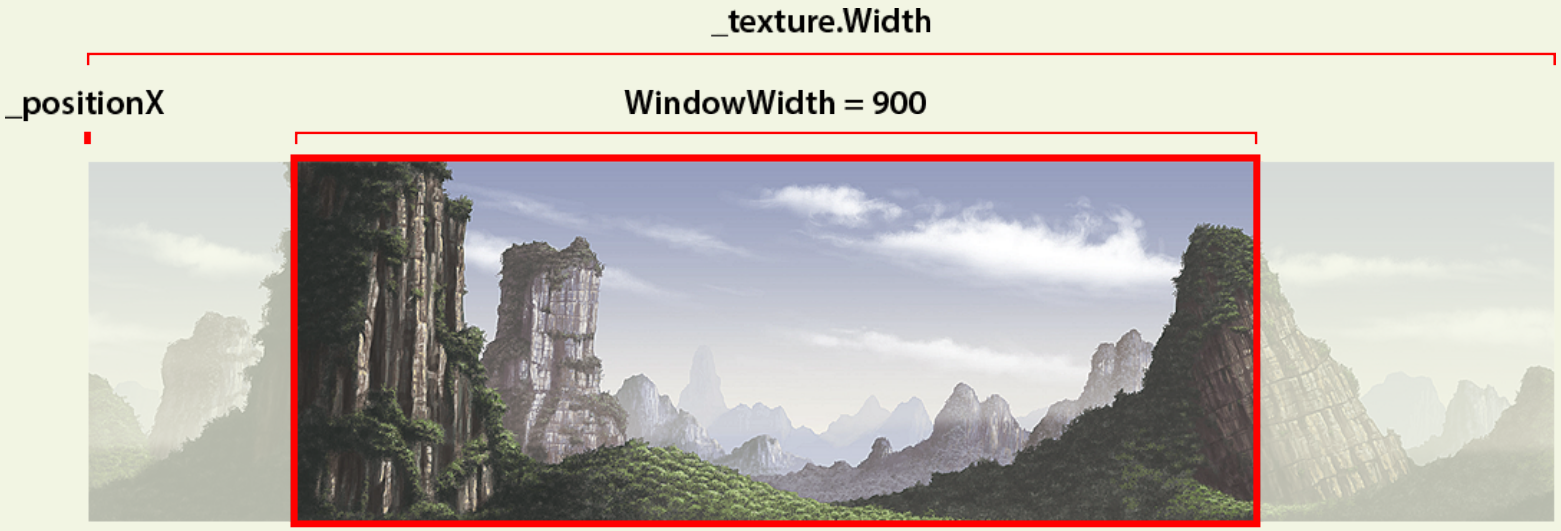
SharpDX

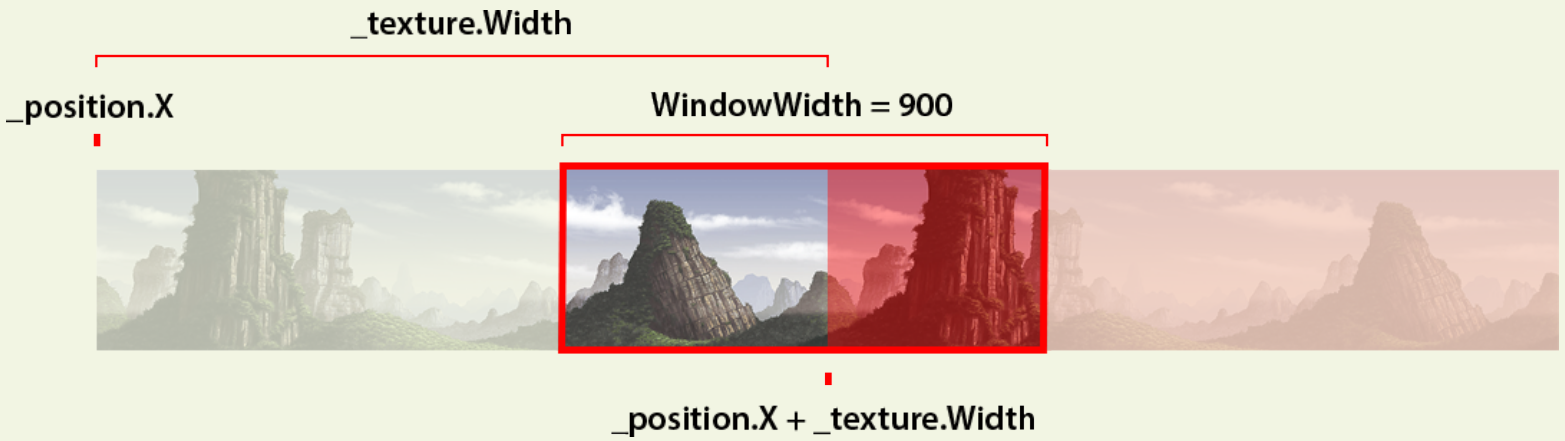
+	-
À jour	Doc?
Tutos	
Forum	

5. Introduction à XNA


0
x
y

```
public class Game : Microsoft.Xna.Framework.Game
{
    readonly GraphicsDeviceManager _graphics;
    SpriteBatch _spriteBatch;

    public Game() { ... }

    protected override void Initialize() { ... }

    protected override void LoadContent() { ... }

    protected override void UnloadContent() { ... }

    protected override void Update(GameTime gameTime)
    {
        // Logique du jeu
    }

    protected override void Draw(GameTime gameTime)
    {
        // Affichage du jeu
    }
}
```


_texture.Width
_positionX
WindowWidth = 900


_texture.Width
_position.X
WindowWidth = 900
_position.X + _texture.Width

TP Git/XNA par équipe