Université Nice SOPHIA ANTIPOLIS

Introduction à la Programmation de Jeux Vidéo

Auteur & Présentation:
Stéphane Lavirotte

Mail: Stephane.Lavirotte@unice.fr
Web: http://stephane.lavirotte.com/
Université de Nice - Sophia Antipolis

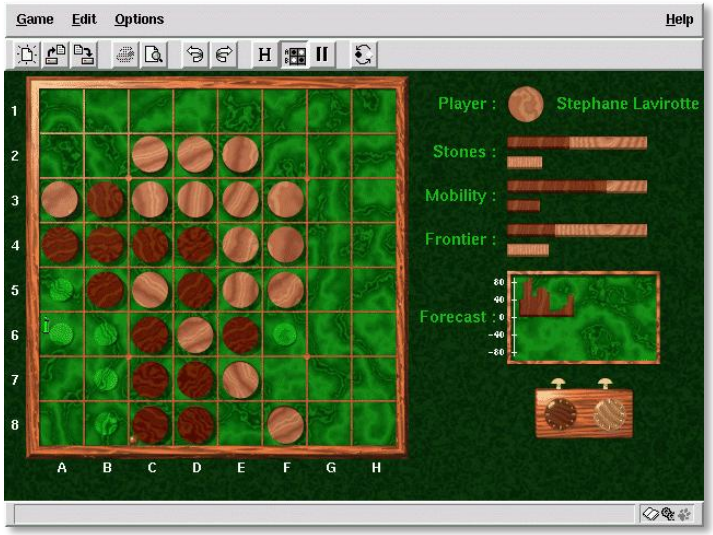
Qui suis-je ?

- ✓ **1995: Ingénieur en Informatique** pour le Génie Logiciel et les Systèmes (Université de Nice – Sophia Antipolis)
- ✓ **2000: Docteur en Sciences pour l'Ingénieur** : Génie Logiciel (INRIA)
- ✓ **2000**: **Post-Doctorat** CNRS (projet européen)
- ✓ **Depuis 2001: Maître de Conférences** en Informatique:
  - A l'IUFM Célestin Freinet (C2i2E)
  - A Polytech'Nice Sophia
    - Départements Sciences Informatiques (parcours Informatique Ambiante et Mobile)
- ✓ Recherche: **Informatique Ambiante**
  - Au Laboratoire I3S (Université de Nice – Sophia Antipolis / CNRS)

Et le jeu dans tout cela ?

Expériences du Jeu Vidéo

- ✓ (Ancien) Joueur
- ✓ Programmeur: Darwersi
- ✓ Encadrement:
  - Ubiquarium et son environnement 3D
  - Projets de jeu pour DeViNT
  - Projets Jeux Pervasifs





Introduction de l'Introduction

Il faut bien commencer par quelque chose !

Introduction

- ✓ Qui programme des jeux vidéo
  - Public de joueur
  - Mais séduit aussi des développeurs non joueurs
- ✓ Idées reçues sur la programmation d'un jeu vidéo
  - Beaucoup pensent que c'est fun comme jouer
- ✓ La réalité
  - Techniques et technologies méconnues
  - Investissement dans des connaissances théoriques
  - Besoin de connaissances pratiques poussées
  - Coder un jeu vidéo ne se révèle pas une partie de plaisir
  - Demande rigueur et de nombreuses compétences

But de ce Cours

- ✓ On ne pourra pas :
  - Faire de vous des dieux de la programmation du jeu vidéo
  - Maitriser toutes les technologies et techniques
- ✓ But :
  - Introduire la programmation du jeu vidéo
  - Orienter et conseiller les débutants
  - Fournir de bonnes bases pour démarrer
  - Pour continuer sereinement le développement
  - Avoir conscience du travail de programmation pour les managers
- ✓ Basé sur la pratique
  - Nous allons essayer de faire des petits jeux avec différents outils
- ✓ Ce cours ne parlera que de programmation !
  - le game design, la scénarisation, le dessin, la modélisation ne seront pas abordés

Programmer un Jeu Vidéo

- ✓ Programmer un jeu vidéo
  - Tâche longue et complexe
  - Nécessite de nombreuses compétences
- ✓ La programmation d'un jeu vidéo
  - De nombreux domaines avec des connaissances théoriques
    - rendu 2D / 3D: algèbre linéaire
    - moteur physique: mathématiques et physique
    - intelligence artificielle: logique
    - réseau: protocoles de bas niveau
    - rendu sonore: traitement du signal
    - scripting: programmation
    - dev pour console: systèmes et programmation embarqués
  - Le motivation ne suffit pas, il faut des compétences ou les acquérir
- ✓ Impossible de tout maitriser !
  - Nécessité de se spécialiser dans un domaine voir un sous domaine
  - Pour devenir un expert

Motivations

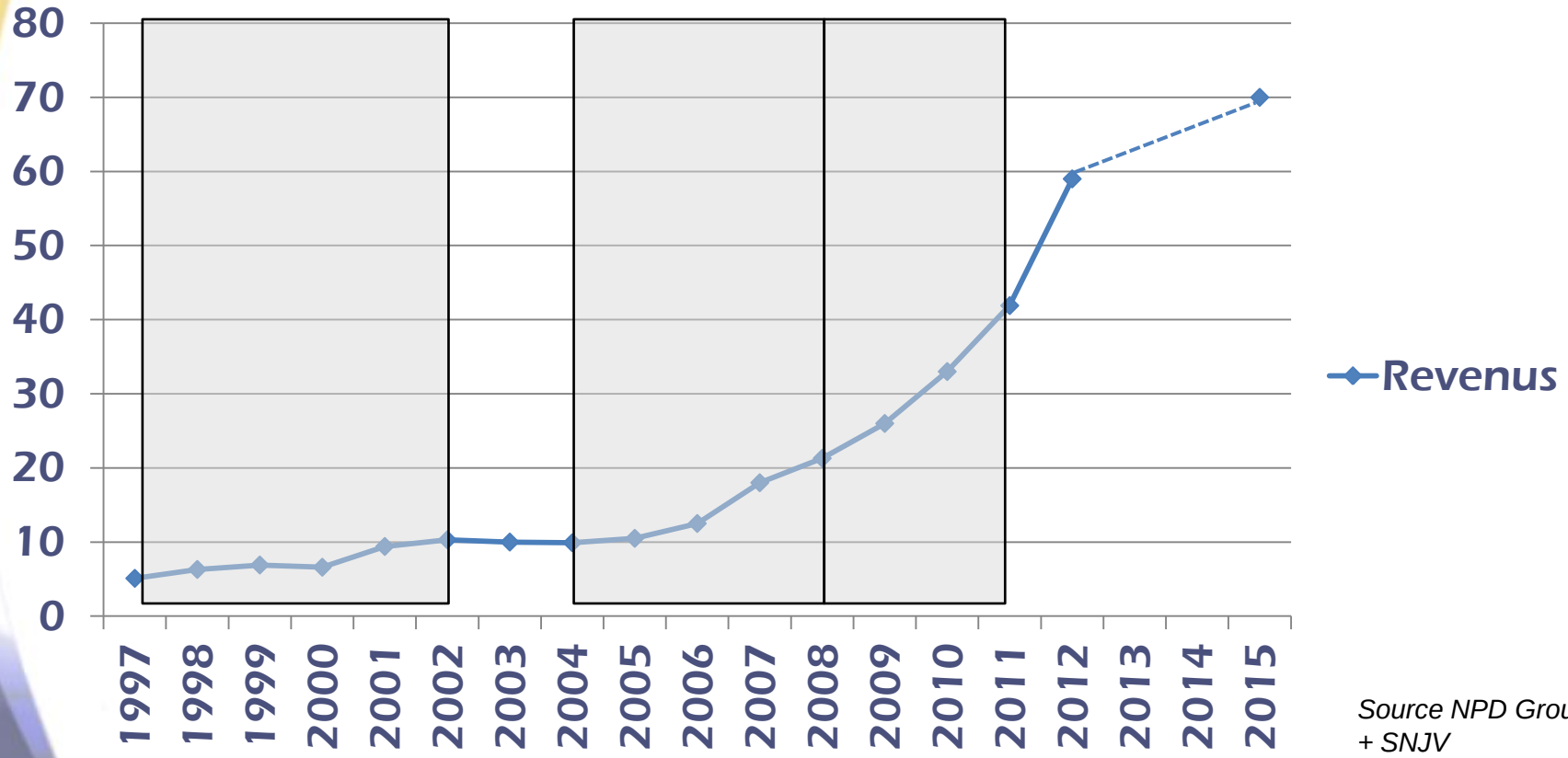
- ✓ Deux types de programmeurs de jeu
  - Concrétiser un esprit créatif dans une production
    - Plus difficile car souvent nécessaire de maitriser plusieurs technologies
  - Coder pour coder
    - Acquérir un maximum d'expérience dans un domaine
    - Quelque soit le jeu support
- ✓ Incidence sur les technologies utilisées

L'Industrie du Jeu Vidéo

Quelques chiffres et analyses

Une Industrie en Croissance…

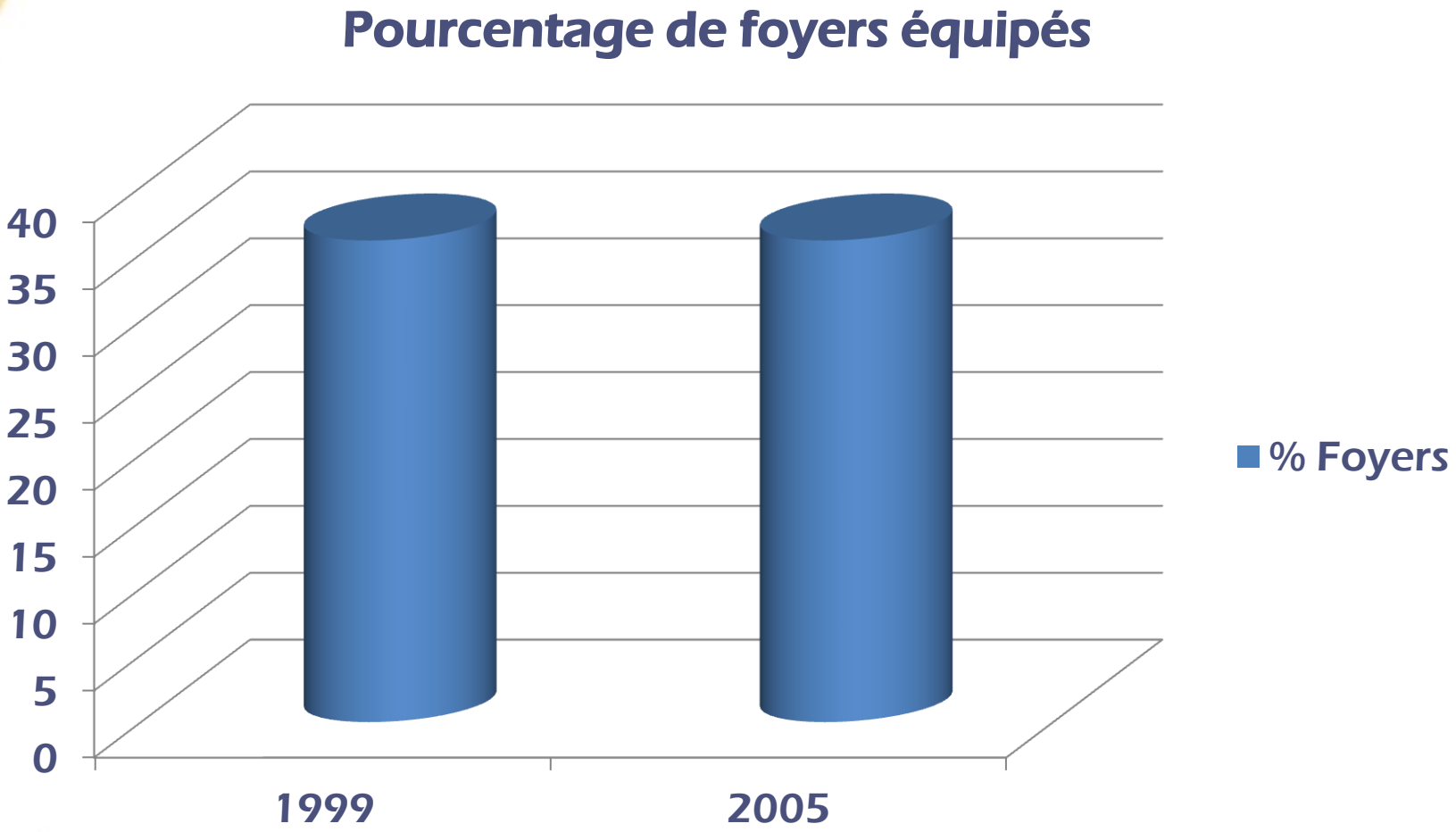
Revenus annuels de l'Industrie du Jeu Vidéo (en milliards de $)



Source NPD Group + SNJV

✓ x2 entre (1997, 2002), (2004, 2008), (2008, 2011)

… Mais sur la même Période …



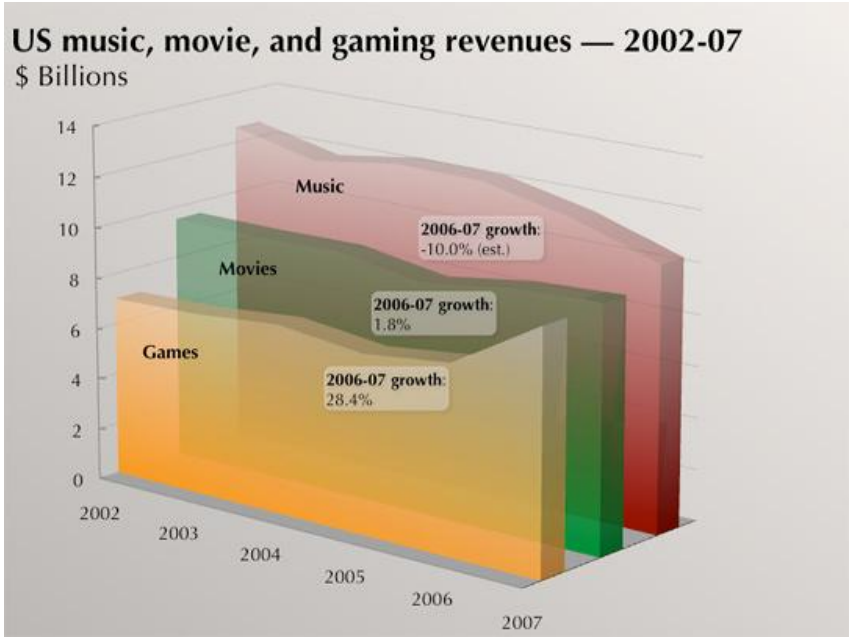
- ✓ Le nombre de foyers équipés stagne
- ✓Donc plus de dépenses pour les mêmes personnes

Le Jeu : un Milieu trop Fermé

- ✓ Fermé au niveau des développeurs
  - Pas assez de nouveaux talents, pas assez de formations
  - Explosion des coûts
  - Suites à répétition, manque de nouveautés
- ✓ Fermé au niveau des consommateurs
  - Trop cher
  - Pas assez personnalisé, produit de masse
  - Intimidant
- ✓ Fermé pour la communauté
  - Pas de moyen d'exprimé sa créativité
  - Difficulté à trouver une audience

Et les Autres Industries ?

- ✓ Film et Musique
  - Ouverture dans la douleur
  - La distribution tend à rester un monologue
  - Vécu comme une menace
  - Mais… création de nouveaux genres
  - Poursuite permanente de l'évolution et plus de maîtrise



- ✓ Arrivées de sites à audience
  - Web 2.0 = C2C, fournisseur d'audience
  - Création de communautés
  - EBay, YouTube, MySpace, etc.

Des Marchés… Nouveaux

- ✓ Des marchés classiques « à saturation » ?
  - Le PC
  - La console de jeu
- ✓ De nouveaux marchés
  - Consoles portables (nomadisme)
    - PSP,
    - Nintendo DS (4+ millions en France, 20+ millions en Europe)
    - …
  - Jeux sur téléphone portable (nomadisme ou mobilité)
  - Accessoires pour interactions plus « naturelle »
    - Nintendo DS: prise en compte du microphone (ex: souffler)
    - Wii et tous ces accessoires…
    - Kinect

Un Public… Nouveau

- ✓ Différents types de joueurs
  - Hardcore Gamer
    - Homme, jeune, célibataire, entre 15 et 25 ans, sur console
  - Joueur occasionnel
    - Jeux sur Internet, tout public
- ✓ De nouveaux publics
  - Public féminin (1/3 des ventes en France)
    - Offre logicielle ciblée
  - Public plus âgé
    - Expérience de jeu intuitive: facilité de prise en main
    - Interaction plus « naturelle » (pas de manette avec 30 boutons !)

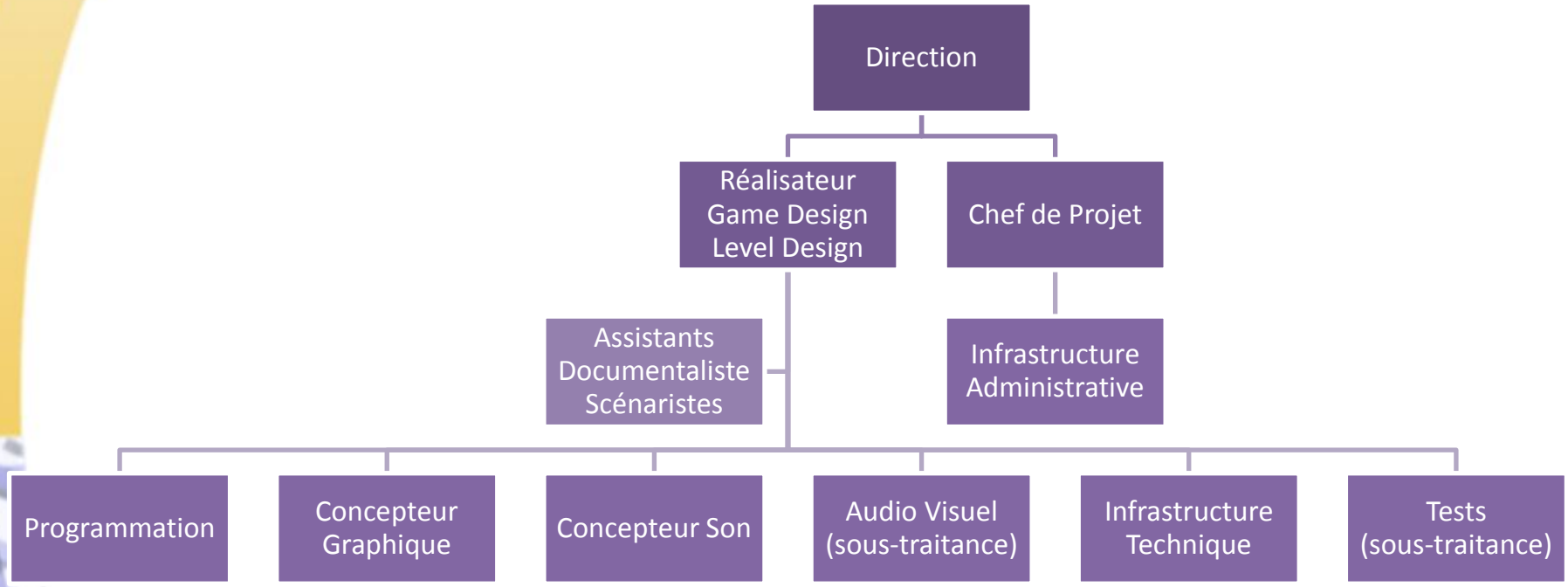
Conclusion partielle

- ✓ Un marché en constante évolution
  - « Mutations » technologiques
  - Conquête de nouveaux marchés
  - ...
- ✓ Sans être l'eldorado, de belles perspectives d'évolution

Un Jeu Vidéo est un Programme

Qui se réalise en équipe

Organigramme



Programmeurs

- ✓ **Programmeurs**
  - **Mission:**
    - Écrire dans un langage spécifique le code informatique permettant de faire évoluer le jeu
    - Fabriquer/Déclencher les images, les sons, ... en fonction des actions du joueur
- ✓ **Deux types de métiers**
  - **Programmeur Jeu**
    - Développe les comportements interactifs de tous les objets du jeu
    - Développe également une interface simplifiée de programmation (langage de script)
    - S'appuie sur les fonctions de base du moteur de jeu
  - **Programmeur Moteur**
    - Développe et adapte les fonctions qui permettent de synthétiser tous les comportements
    - Fournit des versions de ces fonctions pour tous les environnements
    - Plusieurs moteurs: graphique, son, physique, réseau et Intelligence Artificielle
- ✓ La plus grande partie des jeux est développée en C++ à l'exception des jeux sur mobiles qui sont souvent programmés en Java.

Processus de développement

- ✓ Durée approximative: 18 à 24 mois voir 36 mois
- ✓ Coût: entre 300.000 € et jusqu'à 20 millions d'€
- ✓ Personnel: entre 5 personnes sur 8 mois et 100 personnes sur 3 ans
- ✓ Seuil d'amortissement: 100.00 pièces vendues (jeux de consoles et PC)
- ✓ Processus de production lent / durée de vie du produit

Programmer

- ✓ Expliquer à l'ordinateur (via le programme) comment réagir aux commandes du joueur
  - Si il y a un plateau sous mon personnage
    - Faire atterrir le personnage
  - Sinon
    - Le personnage tombe dans l'eau, Game Over
- ✓ Ce qu'il faut programmer
  - L'animation et l'évolution des sons en fonction des actions du joueur
  - Le déplacement des caméras
  - L'intelligence des ennemis, des partenaires...
  - Les communications, ...

Des Questions à se Poser

Des choix à faire...

Proposition de Jeu

- ✓ Quel jeu ?
  - Avoir une idée originale
    - Game Design: Concevoir un univers, un but, des règles
  - Développer cette idée
    - Level Design: Description d'un scénario dans cet univers

- ✓ Avant de se lancer, il faut se poser les questions fondamentales:
  1. Quoi ? Genre: les principales classes de jeu
  2. Pour qui ? Public visé: différents types de joueurs

Les Principales Classes de Jeux

- ✓ Action
  - Un paquet de boutons et de manettes à agiter frénétiquement
- ✓ Aventure
  - L'histoire est essentielle
- ✓ Stratégie
  - Un processus de décision complexe
- ✓ Simulation
  - De l'exercice cybernétique
- ✓ Puzzle
  - Résolution d'une énigme
- ✓ Découverte
  - L'équivalent d'un documentaire

Les Types de Joueurs

- ✓ Un monde divisé en deux catégories ?
  - Les « hardcore gamers »: ceux qui jouent tout le temps
  - Les « casual gamers »: ceux qui jouent de temps en temps

- ✓ Le monde n'est jamais noir ou blanc:
  - « **power gamers** »: passionnés (11% des joueurs pour environ 1/3 des revenus)
  - « **social gamers** »: n'aiment jouer qu'en interaction avec d'autres
  - « **leisure gamers** »: joueur de loisir qui jouent beaucoup mais à des jeux sans importance
  - « **dormant gamers** »: adorent les jeux mais n'y consacrent que peu de temps à cause de contraintes familiales, de travail, ...
  - « **incidental gamers** »: qui ne jouent que s'ils s'ennuient
  - « **occasional gamers** »: joueurs occasionnels qui jouent à des jeux de société, puzzle ou mots croisés en jeu vidéo

D'autres points à valider

- ✓ Les points cruciaux
  - Le « game play » (très incertain)
  - Le design d'interaction utilisateur (innover est une des clés)
  - Le choix graphique (central mais difficile d'innover)
  - L'ambiance sonore (important pour la version finale)
  - L'intelligence artificielle (rejaillit sur le « game play »)
- ✓ Les choix technologiques
  - Pour quelle plate-forme ?
  - Avec quels langages et outils ?
- ✓ Importance d'un prototype
  - Avoir un premier prototype jouable très rapidement
  - Limiter au maximum les fonctionnalités au départ
  - Enrichir le jeu si les sensations de jeu sont bonnes

« Game Play »

- ✓ Bien cibler les objectifs principaux du jeu
  - Limiter les options dans une première version
  - Bien tester les sensations de jeu avant de poursuivre le développement
  - Importance du prototype initial (minimal mais fonctionnel)

- ✓ Le Game Play
  - Point crucial du jeu:
    - Le jeu est-il « fun » ? Le public visé va-t-il accrocher ?
  - Ne se décide pas sur le papier
  - Question d'expérience
  - Besoin de tests pour fixer :
    - le niveau de départ (faciliter la prise en main)
    - la progression de la difficulté (éviter les décrochements)

Interaction

- ✓ Un des points les plus importants
  - Point important pour le « game play »
- ✓ Aucun progrès durant des années
  - Sur console
    - Manettes avec de plus en plus de boutons
    - Quelques innovations: vibration, volant, ...
  - Sur PC
    - Classique: clavier, souris, joystick
- ✓ Une (r)évolution...














Des choix graphiques importants

- ✓ Un jeu sans graphisme possible ?

- ✓ 2D ou 3D ?
  - Quelles sont les exigences liées au jeu ?
  - Compétences dans l'équipe
    - 2D: Dessin, graphisme, photos; 3D: Modélisation, Animation, ...
  - Temps disponible
    - Graphisme 2D plus rapide que 3D

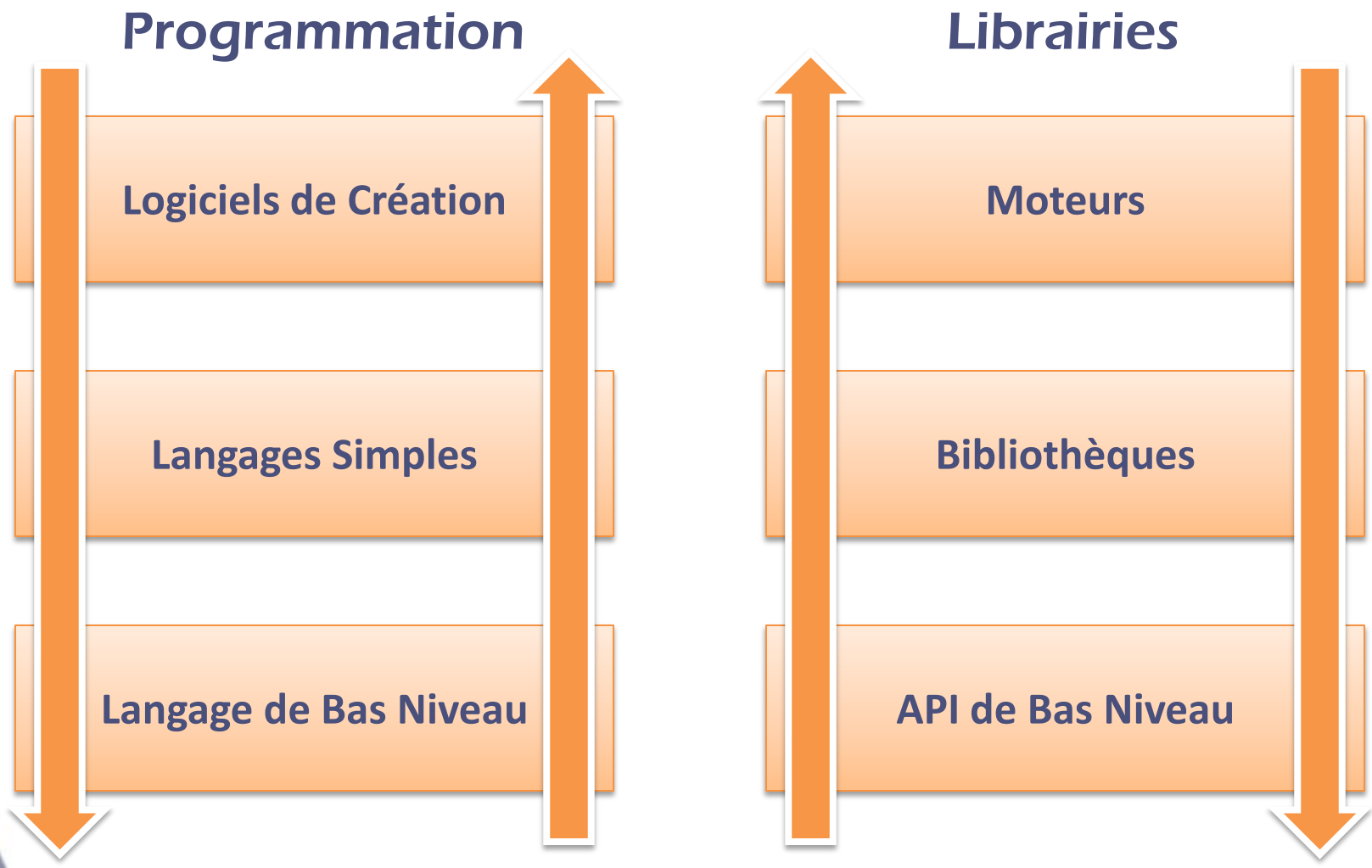
- ✓ Type d'audience
  - Pour quel type d'ordinateur ?
  - Quels sont les outils disponibles sur ces plates-formes ?

Un jeu pour quelle cible ?

- ✓ Cibles
  - Ordinateur personnel
    - Différents systèmes: PC: Windows, Unix, …, MAC: OS X
  - Console de salon
    - Xbox 360, Playstation 3, Wii, …
  - Console portable
    - Game Boy, Nintendo DS, PSP, …
  - PDA, SmartPhone
    - Windows Mobile (CE), Symbian, Linux, …
  - Téléphone portable
    - Propriétaire, Java, Windows Mobile, Symbian, Linux, …

- ✓ Mais le développement se fera toujours sur ordinateur
  - Générer des fichiers pour la cible visée

Quelques outils



Des Logiciels pour la Création

- ✓ RPG Maker
  - Réaliser des RPG en 2D sans programmer
  - Multiples versions toutes payantes
- ✓ Mugen
  - Réaliser des jeux de combat 2D avec peu de programmation
  - Logiciel multi-plateformes (Windows, DOS, Linux) gratuit
- ✓ 2D Fighter Maker
  - Proche de Mugen
- ✓ The Game Factory, Multimedia Fusion
  - Outils de création de jeu sans programmation (ou très peu)
- ✓ FPS Creator
  - Création de jeux 3D

Des Langages Simples

- ✓ DarkBasique Pro
  - Environnement de programmation payant
  - Orienté jeu
  - Langage assez simple
- ✓ PureBasic
  - Environnement de programmation 2D et 3D payant
  - Portable: Windows, Linux, MacOS, AmigaOS
- ✓ BlitzMax
  - Environnement complet
  - Syntaxe simple
- ✓ Hyperion
  - Simplicité et puissance
  - Sous Windows uniquement

Des Langages pour Programmer

- ✓ Des familles de langages
  - Langage de bas niveau
    - Assembleur, C, ...
  - Langages Objet
    - C++, Java, C# ...
  - Langages de script
    - Python, Scheme, ...
- ✓ Avec des environnements de programmation
  - Java: Eclipse
  - C++, C#: .NET
  - ...

Des Moteurs

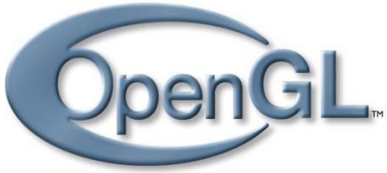
- ✓ Moteur Graphique
  - Gère la création et l'affichage de la scène,
  - L'optimisation des rendus, les effets spéciaux,
  - Le chargement et la sauvegarde des modèles, ...
- ✓ Moteur Physique
  - Gère l'apesanteur, les collisions,
  - Le comportement des objets, des fluides, ...
- ✓ Moteur Réseau
  - Gère les communications
  - Performances optimales
- ✓ Moteur Audio
  - Gère le chargement, la lecture
  - Gère le son 3D et toute sorte d'effets spéciaux

Des Bibliothèques

- ✓ Bibliothèque (Library) est un ensemble de fonctions
  - permettant d'effectuer des actions spécifiques
  - Regroupées en une ou plusieurs entités
- ✓ Exemple de Bibliothèques
  - kjAPI (C++)
  - SxDL (C++)
  - ClanLib (C++)
  - Allegro (C / C++)
  - Artificial Engines (.NET)
  - RealmForge GDK (.NET)
  - XNA (C# .NET)
  - LWJGL (Java)
  - Pygame (Python)

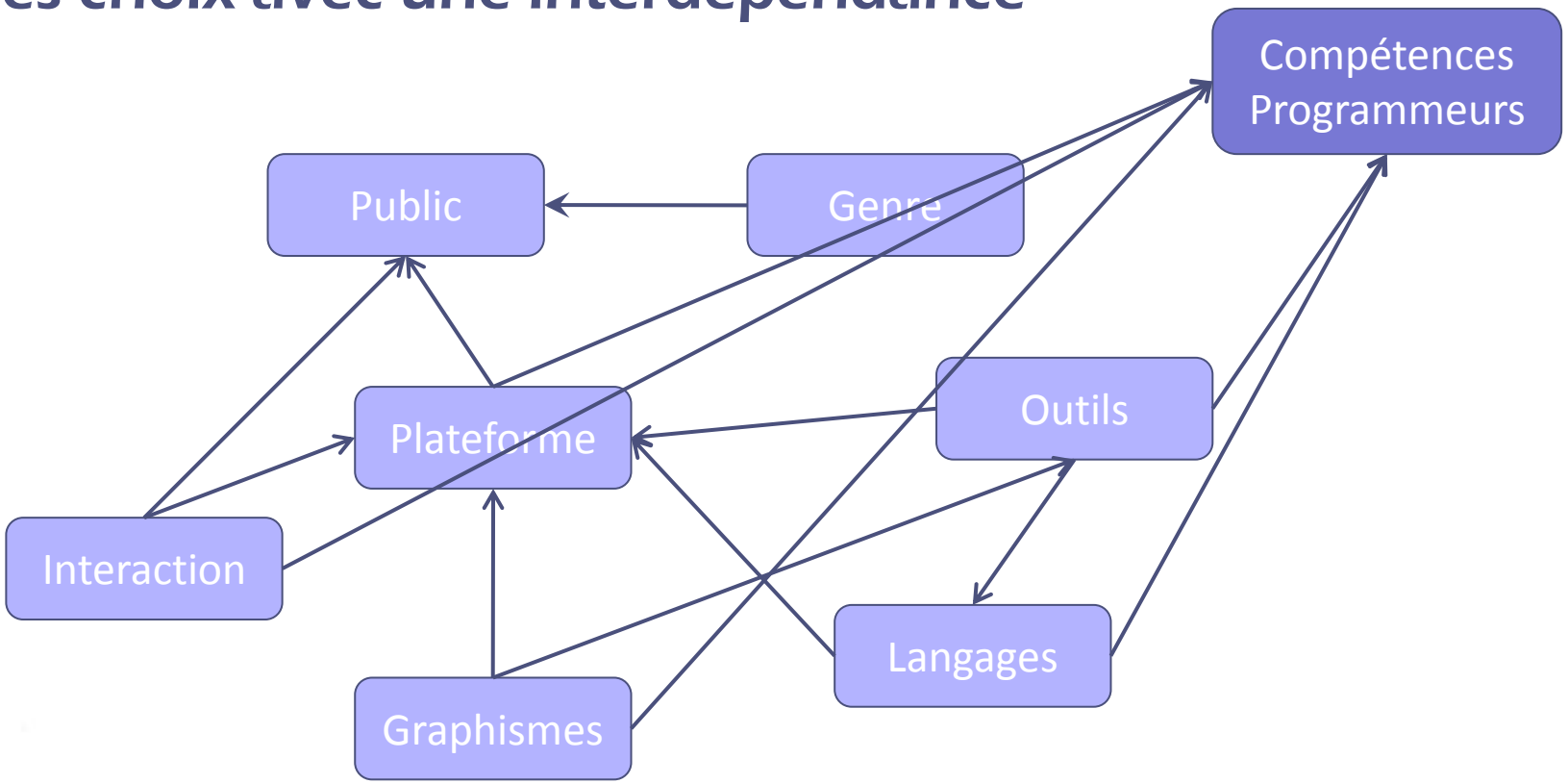
Des API de bas niveau

- ✓ Graphisme 2D et 3D
  - DirectX (Direct3D)
  - OpenGL
  - SDL
- ✓ Réseau
  - TCP/IP
  - UDP
- ✓ Audio
  - DirectMusic
  - OpenAL
- ✓ Gestion des périphériques
  - DirectInput



Conclusion

- ✓ Une bonne idée de jeu ne suffit pas !
- ✓ Des choix avec une interdépendance



Informatique et Programmation

Implémentations:
Approche Structurée
Approche Objet

Qu'est ce que l'Informatique ?

- ✓ Sémantiquement:
  - Contraction des termes: information et automatique
  - Science étudiant le traitement automatique de l'information (l'ordinateur n'est qu'un cas particulier de l'informatique)
- ✓ Information:
  - **Théorie de l'Information**: théorie visant à quantifier et qualifier la notion de contenu en information présent dans un ensemble de données
- ✓ Traitement Automatique:
  - **Algorithmique**: ensemble des règles et techniques impliquées dans la définition des suites d'actions (opérations) à effectuer pour résoudre un problème
- ✓ Théorie de la programmation:
  - **Génie Logiciel**: ensemble des activités qui permettent l'écriture des programmes informatiques

Un Algorithme C'est Quoi ?... Une Recette de Cuisine ?

- ✓ Algorithme:
  - Méthode systématique de procéder pour faire quelque chose
  - Concept applicable mécaniquement, **sans réfléchir**, en suivant simplement un mode d'emploi précis
- ✓ Donc un algorithme est « presque » comme une recette de cuisine
  - Exemple du quatre-quarts (même les garçons peuvent faire cette recette)
    - prendre deux œufs, le même poids de farine, de beurre et de sucre,
    - ajouter le parfum,
    - mélanger le tout,
    - mettre au four une petite demi-heure
  - Et si cette recette était faite par une intelligence mécanique ?

Quels Eléments pour Ecrire un Algorithme ?

- ✓ 5 éléments de bases à enseigner:
  - **Séquence d'instructions**: décrire une succession d'opérations à effectuer
  - **Instruction conditionnelle**: le test pour exprimer des choses à faire en fonction de conditions
  - **Boucle d'instructions**: Répéter une opération n fois ou jusqu'à ce qu'une condition soit vraie
  - **Variables**: symbole utilisé pour marquer un rôle dans un algorithme (renvoie à une position de mémoire dont le contenu peut prendre différentes valeurs pendant l'exécution)
    - **Affectation**: donner la valeur d'une expression à une variable
  - **Fonctions**: regrouper un bloc d'instructions
- ✓ Un élément supplémentaire pour les langages OO
  - Classe: regrouper des fonctions

Techniques de Programmation

- ✓ Paradigmes de programmation
  - Programmation impérative
    - Programmation structurée
    - Programmation orientée objet
    - Programmation orientée composant
    - Programmation orientée aspect
  - Programmation déclarative
    - Programmation descriptive
    - Programmation fonctionnelle
    - Programmation logique
  - Programmation par contrat
  - Programmation par contraintes
  - Programmation concurrente
  - Programmation procédurale
  - Programmation par intention

Programmation Structurée

- ✓ Sous ensemble de la programmation impérative
- ✓ Paradigme important de la prog. apparu en 1970
  - Article fondateur de *Dijkstra* dans *Communications of the ACM* nommé « *GO TO statement considered harmful* »
- ✓ Volonté de supprimer les *goto*
  - Sauf pour les cas exceptionnels (exception)
- ✓ Souvent utilisé conjointement à la méthodologie de décomposition successive (approche top-down)
  - Décomposition de la structure à large échelle d'un programme
  - Opérations plus petites
  - Implémente et tester ces opérations « élémentaires »
  - Les assembler pour réaliser un programme

Programmation Objet

- ✓ Sous-ensemble de la programmation impérative
- ✓ Consiste en la définition et l'assemblage de briques logicielles appelées *objet*
  - Un objet représente un concept, une idée ou toute entité du monde physique
- ✓ Un objet est une structure de données valuées et qui répond à un ensemble de messages
  - Cette structure de données définit son **état**
    - Les données ou champs qui décrivent sa structure interne sont appelées ses **attributs**
  - L'ensemble des messages qu'il comprend décrit son **comportement**
    - L'ensemble des messages forme ce que l'on appelle **l'interface** de l'objet, utilisé pour faire interagir les objets entre eux

Université Nice SOPHIA ANTIPOLIS

Et Pourquoi Python ?

- ✓ Python
  - Langage simple à lire et écrire
  - Facile à apprendre
  - De nombreux tutoriel et codes exemple
  - Raisonnablement rapide
  - Langage de script (peut être embarqué dans un prog C ou C++)
- ✓ Pygame
  - Fournit une librairie pour faciliter la gestion de l'interface utilisateur, l'affichage, la gestion du son, ...
  - Uniquement 2D
- ✓ PyOpenGL
  - Offre à Python la puissance d'OpenGL pour des performances 3D
- ✓ DirectPython
  - Fournit à python l'accès à l'interface DirectX (9.0c), incluant Direct3D, DirectSound, DirectShow and DirectInput

Python / pygame

- ✓ Python
  - Langage de Programmation Orienté-Objet Dynamique
  - Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm et téléphone Nokia
  - Porté sur les environnement .NET et Java
  - Licence Open Source
- ✓ PyGame
  - Modules Python pour l'écriture de jeux
  - Basé sur le librairie SDL (Simple Directmedia Layer)
  - Windows(95, 98, me, 2000, XP, Vista, 64bit, etc), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, et QNX

Structure de Données

- ✓ **Chaîne de caractères**
  - Entre simple ou double quote
  - `x = 'Hello world !'; y = "Hello world ! "`
- ✓ **Liste**
  - `a = ['hello', 'world', 1, 12, 123, 1234]`
  - `a[0] >>> 'hello'`
  - `len(a) >>> 6`
  - `q = [2, 3]`
  - `p = [1, q, 4] >>> [1, [2, 3], 4]`
  - `p[1] >>> [2, 3]`
  - `p.append(5) >>> [1, [1, 2, 3], 4, 5]`
  - `p[1][0] >>> 2`

Structures de Contrôle

- ✓ **Test:**

```
if test:        ( ==, <=, >, …)
    …
elif test:
    …
else:
    …
```

- ✓ **Boucles:**
  - **for** *var* **in** *list*:
    …
  - **while** *test*:
    …
  - **range**(10) >>> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  - **break**
  - **continue**
  - **pass**

Classes et Fonctions

- ✓ **Fonction:**
  - `def maFonction(param1, param2 = val):`
    `…`
- ✓ **Classe:**
  - `class MaClasse:`
    `…`
  - `class MaClasse(herit):`
    `def __init__(self, params):`
    `…`
- ✓ **Main:**
  - `def main():`
    `…`
  - `if __name__ == "__main__": main()`

Py2exe: Créer un Exécutable

- ✓ Installation de py2exe
  - http://www.py2exe.org/
- ✓ Créer un script setup.py

```
from distutils.core import setup
import py2exe
setup(console=['helloworld.py'])
```

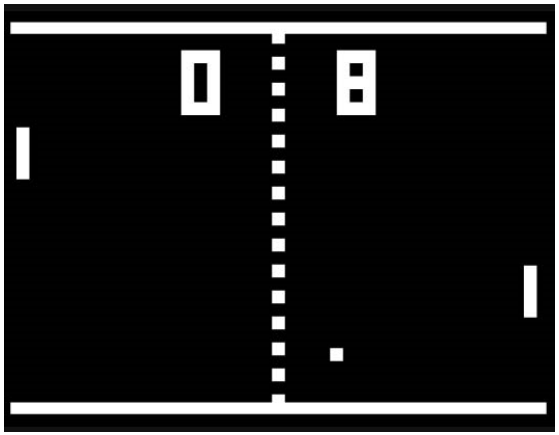
- ✓ Exécuter le script setup.py

```
python setup.py py2exe
```

- ✓ Tester l'exécutable créé:

```
cd dist
helloworld.exe
```

- ✓ Ce qui est généré peut être utilisé par un « installeur »



« Mon » Premier Jeu

PyPong: Pong en Python

Qu'est ce qu'un Jeu Vidéo ?

- ✓ Jeu Vidéo
  - Application « multimédia » (image, vidéo, son, ...)
  - Interactif (si ça joue tout seul, c'est moins drôle !)
  - Souvent gourmand en puissance de calcul
  - Nécessitant une programmation optimisée

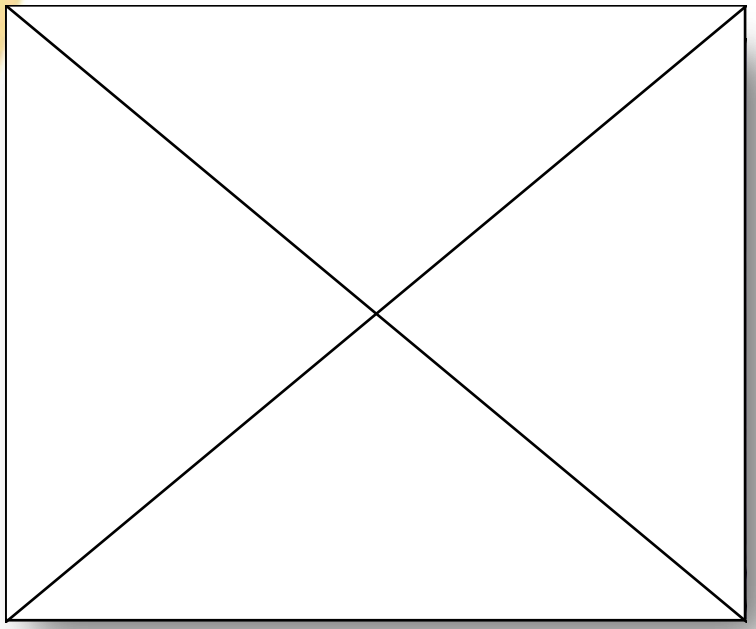
- ✓ Les paramètres à gérer (avancement du jeu)
  - Gestion du joueur (événements)
  - Gestion des personnages, objets mobiles, collisions, ...
  - Gestion de l'environnement (météo, lumière, son)
  - Paramètres du jeu (argent en caisse, consommation des ressources, ...)
  - Etc.

Histoire de Pong

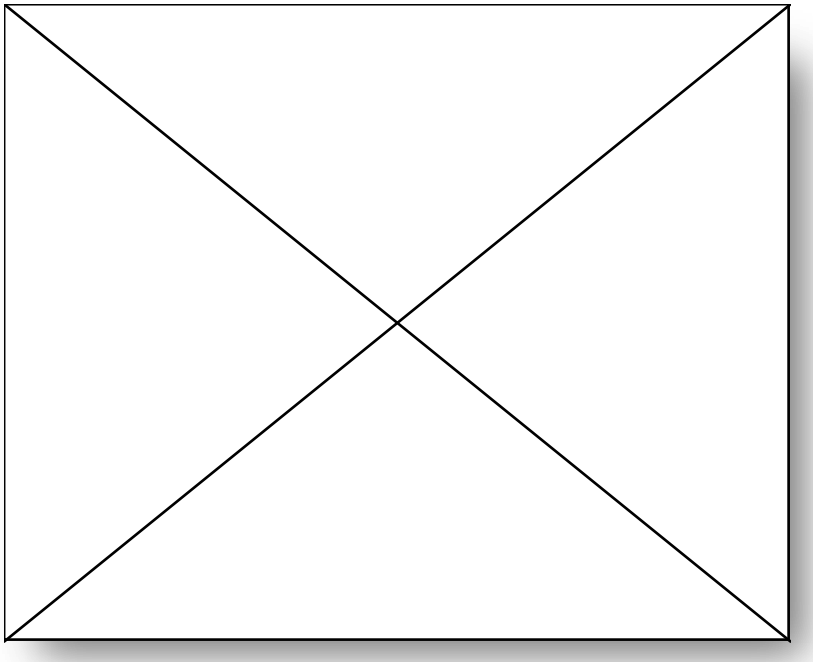
- ✓ Pong
  - Jeu vidéo inspiré du tennis de table
  - Premier jeu vidéo populaire
  - Développé par Atari Inc.
  - Sorti en 1972 pour borne d'arcade
  - Sorti en 1975 sur console de salon
  - Installé dans un bar de Sunnyvale
  - Hors service au bout de quelques jours
    - Le monnayeur est plein et bloque le système !

- ✓ Nous allons recréer Pong pour notre premier jeu !
  - Très simple
  - Permet de voir les bases exposées
  - Sans avoir besoin d'un haut niveau de programmation

Des Versions Plus ou Moins Evoluées

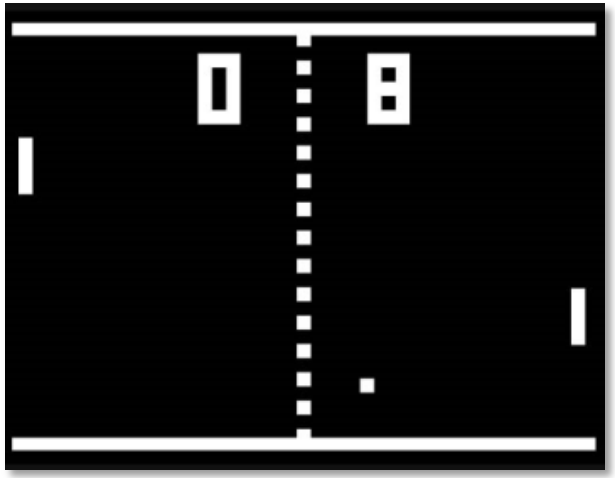


La toute première version



La version borne d'arcade

Pong



- ✓ Composants du jeu:
  - Un terrain: un rectangle + une ligne
  - Deux raquettes: 2 barres verticales
  - Une balle: 1 cercle
- ✓ Règles du jeu
  - La balle rebondit sur les bords du terrain et les raquettes
  - Les raquettes bougent de haut en bas uniquement
  - Si la balle n'est pas renvoyée par la raquette, le joueur adverse marque un point
  - La partie dure jusqu'à ce que le premier joueur marque 10 points
- ✓ Gestion
  - Doit générer le score du match

Squelette de Jeu

- ✓ Initialisations
- ✓ Boucle d'exécution
  - Gestion des évènements
  - Gestion du temps
  - Gestion des déplacements
  - Gestion des collisions
  - Affichage
- ✓ Fermeture propre du jeu

Les cycles s'enchaînent

Ordre des actions peu importante

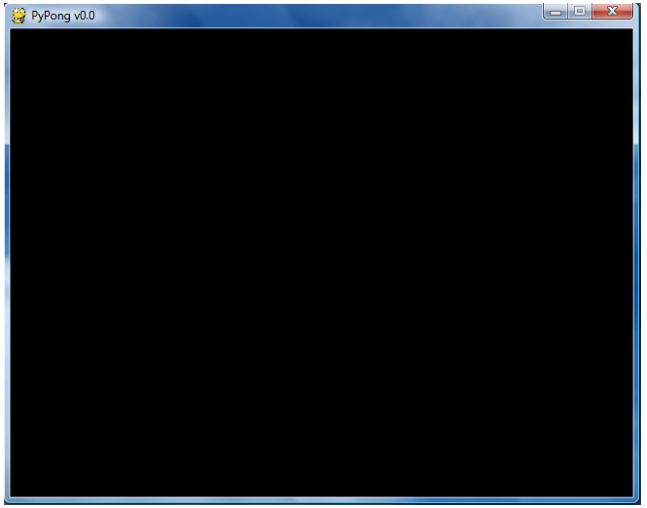
Initialisations

- ✓ **Initialisation de la librairie PyGame**
  - **Chargement des librairies nécessaires**

```
import sys
import pygame
from pygame.locals import *
```

  - **Initialisation de la librairie PyGame**

```
pygame.init()
```



- ✓ **Initialisation de l'application graphique**
  - **Création d'une fenêtre de taille 640 x 480**

```
fenetre = pygame.display.set_mode((640, 480))
```

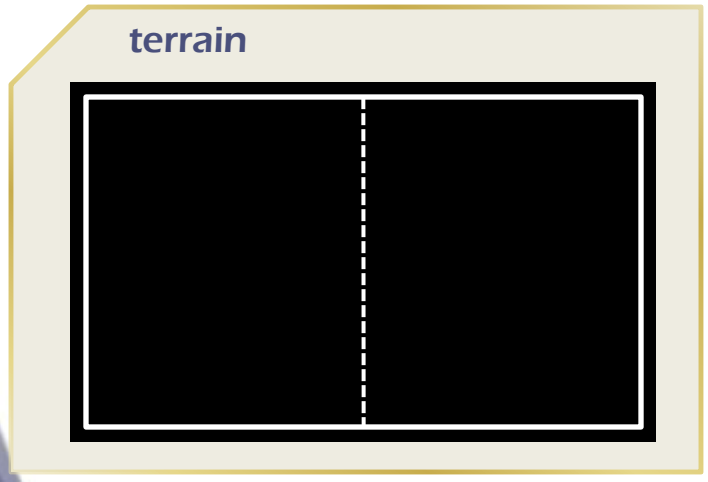
  - **Ajout d'un titre à la fenêtre**

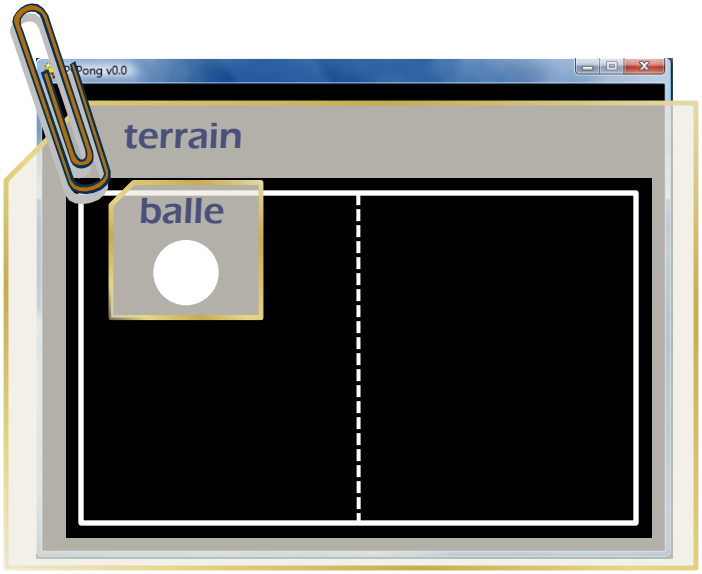
```
pygame.display.set_caption("PyPong v0.0")
```

Un Calque c'est Quoi ?

- ✓ Dessiner des objets sur des calques, ce qui permet:
  - De les bouger facilement
  - De détecter les collisions

- ✓ Avec PyGame
  - Utilisation du concept (objet) Surface
  - Un calque pour chacun des éléments





Création des Eléments Graphiques

- ✓ Création des graphiques: le terrain
  - Créer quelques variables utiles

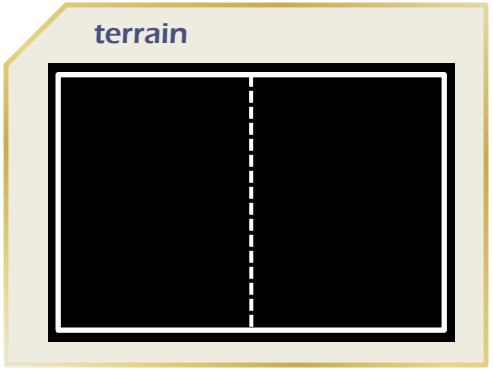
```
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

  - Créer un calque

```
terrain = pygame.Surface(fenetre.get_size())
```

  - Dessiner sur le calque

```
terrain.fill(BLACK)
pygame.draw.rect(terrain, WHITE, Rect((5,5), (630,470)), 2)
pygame.draw.aaline(terrain, WHITE, (330,5), (330,475))
```



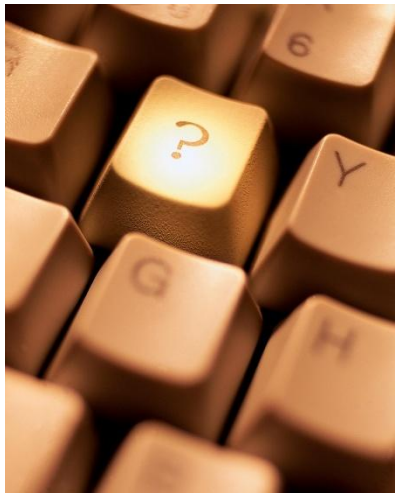
- ✓ Création des graphiques: la balle
  - Approche identique
  - A vous de jouer...

- ✓ Initialisation des sons, images, vidéos, ...

Gestion des Evènements

- **Boucle de traitement des évènements:**

```
 for event in pygame.event.get():
     if (event.type == QUIT) or (event.type == KEYDOWN and
event.key == K_ESCAPE):
         pygame.quit() ; sys.exit(0)
     if event.type == KEYDOWN:
         if event.key == K_UP:
             pass # Mouvement vers le haut
         elif event.key == K_DOWN:
             pass # Mouvement vers le bas
     elif event.type == KEYUP:
         if event.key == K_UP or event.key == K_DOWN:
             pass # Arrêter le mouvement de la raquette
```



- **Traite tous les évènements en attente**

Objets en mouvement

- ✓ Déplacer à chaque frame
  - La balle et les raquettes (Pong)
  - le personnage en 2D dans un jeu d'aventure
  - une voiture en 3D dans un jeu de course
- ✓ Même si cela semble différent, la manière est identique

- ✓ Maîtrise du déplacement
  - Chaque objet a un paramètre de déplacement
  - Processeurs aux performances différentes
    - Ralentir les processeurs rapides ?
  - Cycles inégaux
- ✓ Les déplacements sont sujets aux ralentissements divers et non maîtrisables

Une Horloge

- ✓ Calcul de la durée du cycle précédent
  - Delta_time = get_time – total_time;
  - Total_time += delta_time;
- ✓ Paramètre vitesse plutôt que déplacement
  - Déplacement = Vitesse x Temps

- ✓ Ainsi un même objet se déplace à la même vitesse
  - Quelque soit le processeur
  - Quelques soient les circonstances

Gestion du Temps

✓ Calcul du Temps avec PyGame

- Initialisation d'une horloge

```
clock = pygame.time.Clock()
```

- Dans la boucle de traitement principale

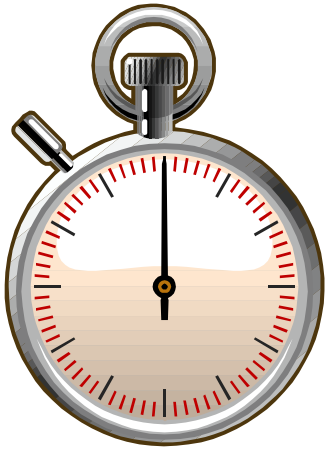
```
while True:
    ...
```

- On compte le temps écoulé réellement

```
time_passed = clock.tick(30)
time_sec = time_passed / 1000.0
```

- On calcule la nouvelle position d'un objet:
  - Nouvelle position = ancienne position + vitesse x temps

```
x += speed_x * time_sec
y += speed_y * time_sec
```

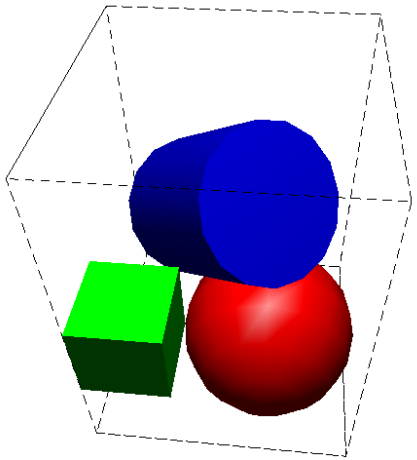


Gestion du Jeu

- ✓ Gestion des éléments du jeu
  - Des variables (ou attributs) sont nécessaires pour stocker l'information
- ✓ Plusieurs éléments à gérer:
  - Terrain
    - dimensions: width, height
  - Balle
    - position: x, y; speed (*en x et en y*)
  - Raquette / Joueur
    - position: x, y; move; score
  - Raquette / Ordinateur
    - Position: x, y; move; speed (*en x et en y*); score

Gestion des collisions

- ✓ Qu'est ce qu'une collision ?
  - Entre deux objets
  - Comment le détecter ?
    - Calcul sur les contours
    - Besoin d'approximation pour les formes « complexes »:
      - boîte englobante



- ✓ Calcul
  - des collisions
    - Balle avec les bords du terrain
    - Balle avec les raquettes
  - du rebond de la balle
    - Inverser l'angle de la balle en fonction du blocage

Affichage

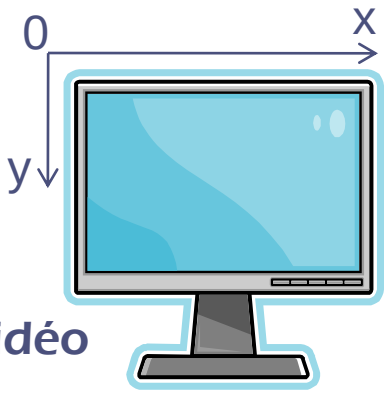
- ✓ Indépendant
- ✓ Pas indispensable
- ✓ A chaque cycle, la caméra montre l'état du jeu
- ✓ Chaque élément du jeu s'affiche lui-même
- ✓ Exemple de code dans le cas général:

```
void display() {
    terrain_display();
    objects_display();    // bâtiment,  végétation
    characters_display();// personnages
    hud_display();        // indicateurs jeu
    if (pause)
        pause_display();
}
```

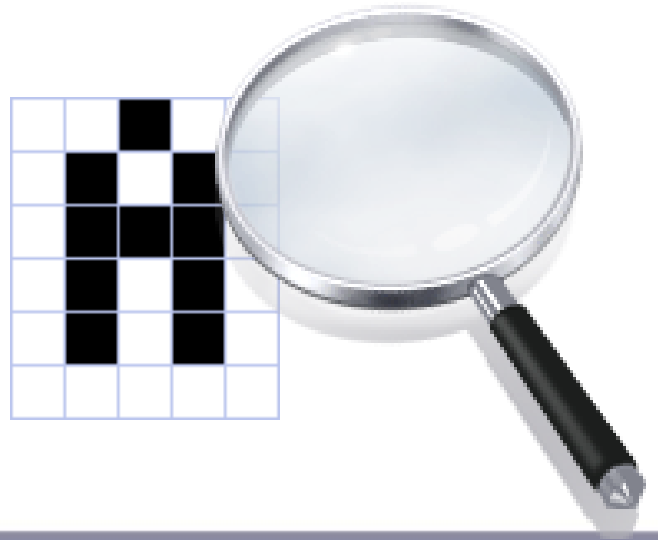
- ✓ Attention l'ordre d'affichage a son importance

Afficher sur un écran: le Pixel

- ✓ Technologie d'affichage
  - L'image s'affiche sur un écran (ou moniteur)
  - Système de coordonnées
  - Chaque point est un élément phosphorescent
  - Point dénommé pixel (PICture ELement)
    - Elément minimal adressable par le contrôleur vidéo
    - Rectangulaire, approximativement carré



- ✓ Image = ensemble de pixels
  - Tableau 2D de pixels



- ✓ Nous verrons la suite au prochain cours !

Gestion de l'Affichage pour notre Jeu

- ✓ **Affichage du terrain**

```
fenetre.blit(terrain, (0, 0))
```

- ✓ **Affichage du score**

```
score1 = font.render(str(raquette1_score),  True, WHITE)
score2 = font.render(str(raquette2_score),  True, WHITE)
screen.blit(score1, (250, 210))
screen.blit(score2, (380, 210))
```

- ✓ **Affichage des objets**

```
fenetre.blit(raquette1, (raquette_x, raquette_y))
fenetre.blit(raquette2, (raquette_x, raquette_y))
fenetre.blit(balle, (balle_x, balle_y))
```

- ✓ **Affichage de la scène construite**

```
pygame.display.update()
```

Gestion du son

- ✓ Son:
  - Effets sonores:
    - Renforce le réalisme (tir sans son)
    - Permet d'aider le joueur (retour autre que visuel)
  - Musique:
    - Permet de créer une ambiance sonore (stress, joie, ...)
- ✓ Utilisation de bibliothèques
  - Gain de temps et résultat rapide
  - Attention aux licences !
- ✓ Quelques pointeurs
  - http://www.soundsnap.com/
  - http://www.universal-soundbank.com/
  - http://www.musique-libre-de-droit.fr/

Gestion du Son dans notre Jeu

- **Initialisation du son :**

```
class NoneSound:
    def play(self):
        pass

if not pygame.mixer:
    print "Warning, sound disabled"
    sound = NoneSound()
else:
    try:
        sound = pygame.mixer.Sound("tennis_ball_bounce.ogg")
    except pygame.error, message:
        print "Cannot load sound"
```

- **Jouer le son :**

```
sound.play()
```

Glossaire

Des fonctions utiles pour le TD

Glossaire de Commandes pour les Travaux Dirigés

- ✓ **pygame.init()**
  - Initialize all imported pygame modules
- ✓ **pygame.display.set_mode(***resolution=(0,0), flags=0, depth=0* **)**
  - Initialize a window or screen for display
- ✓ **pygame.display.set_caption(***title, icontitle=None***)**
  - Set the current window caption
- ✓ **pygame.display.update(***rectangle=None***)**
  - update portions of the screen for software displays
- ✓ **pygame.Surface(***(width, height), flags=0, depth=0, masks=None***)**
  - Pygame object for representing images
- ✓ **pygame.draw.rect(**Surface, color, Rect, width=0**)**
  - Draw a rectangle shape
- ✓ **pygame.draw.circle(***Surface, color, pos, radius, width=0***)**
  - Draw a circle around a point
- ✓ **pygame.event.get()**, **pygame.event.get(***type***)**, **pygame.event.get(***typelist***)**
  - Get events from the queue
- ✓ **pygame.time.clock()**
  - Create an object to help track time
- ✓ **pygame.font.Font(***filename, size***)**
  - Create a new Font object from a file
- ✓ **pygame.mixer.Sound(**filename**)**, **pygame.mixer.Sound(**buffer**)**, **pygame.mixer.Sound(**object**)**
  - Create a new Sound object from a file
- ✓ **Surface.blit(***source, dest, area=None, special_flags = 0***)**
  - Draw one image onto another
- ✓ **Font.render(***text, antialias, color, background=None***)**
  - Draw text on a new Surface