L'informatique et les jeux

L1

Elisa Fromont

8			4		6			7
						4		
	1					6	5	
5		9		3		7	8	
				7				
	4	8		2		1		3
	5	2					9	
		1						
3			9		2			5

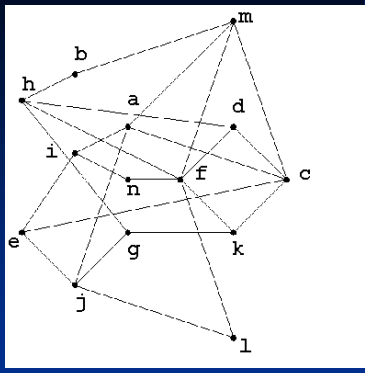
Qui suis-je ?

- Maître de Conférences en Informatique rattachée au laboratoire Hubert-Curien

- Domaines de recherche
  - Apprentissage automatique sous contraintes
  - Fouille de données
  - Programmation logique inductive
  - Bases de données inductives

- Comment me joindre ?
  - par mail : elisa.fromont@univ-st-etienne.fr
  - au labo Hubert Curien: bureau F159
  - À la fac de sciences (département informatique)
  - et évidemment pendant les cours !

Une structure de données utile : le graphe !



- Vous avez dit « structure de données » ???
  - En informatique, *une **structure de données** est une structure logique destinée à contenir des données, afin de leur donner une organisation permettant de simplifier leur traitement.*
- Des exemples de structures de données :
  - Singletons (constantes, variables, ensembles, ….)
  - Tableaux
  - Listes (cf. partie sur les machines abstraites)
  - Arbres (cf. partie sur les arbres)
  - Graphes (une généralisation des arbres)
- Pb : comment manipuler les données dans ces structures ? (= comprendre leur structure logique)

Définitions sur les graphes (1)

- (Graphe)

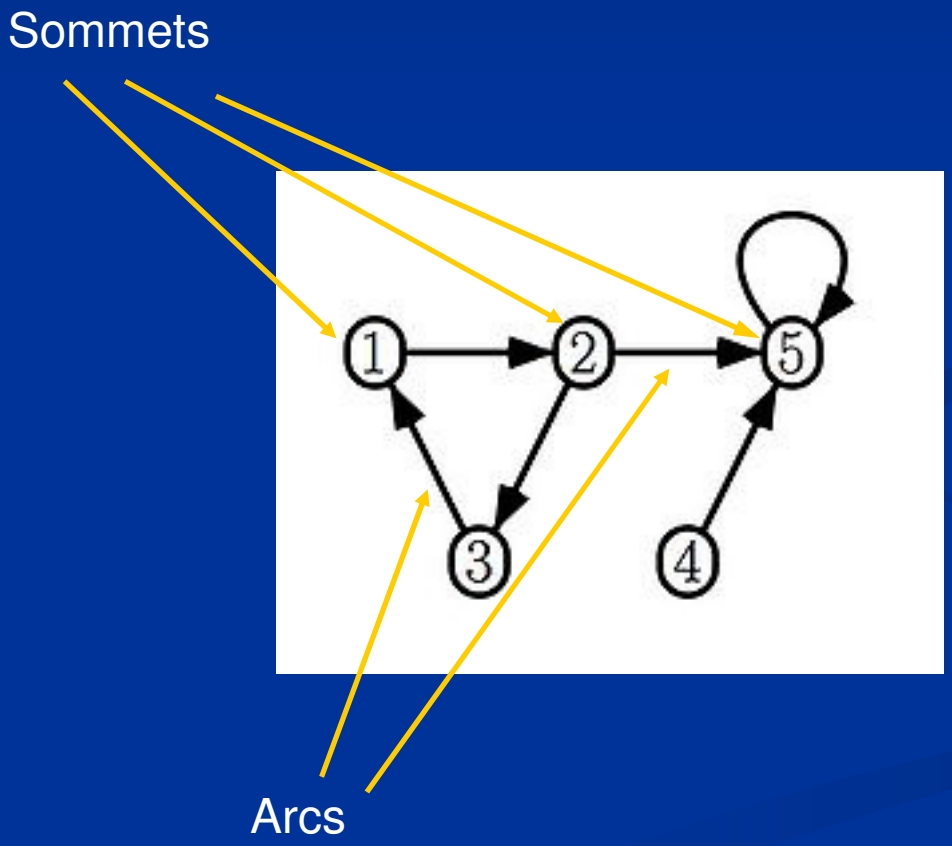
- (Graphe orienté)

Définitions sur les graphes (2)

- (Successeurs et prédécesseurs d'un sommet)

- (Ciruit)

Exemple



$\Gamma+(2)$ =

$\Gamma-(5)$ =

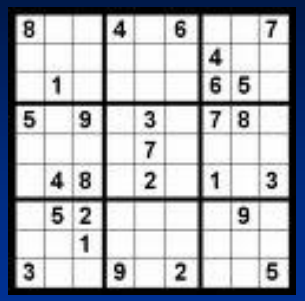
$\Gamma-(4)$ =

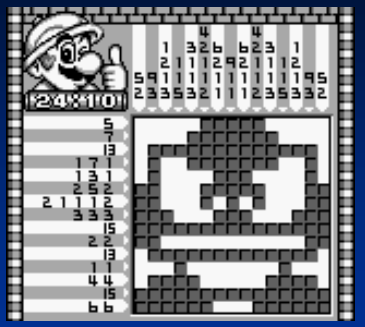
Circuit ?

Ce dont on va parler…

1. Casse têtes (Sudoku, picross…)
   - Programmation par contraintes (pose des contraintes en TD)
2. Jeux de stratégies classiques (Echecs, Dames, Othello, Go…)
   - IA et MinMax (déjà vu en TD sur les arbres)
   - Jeux de Nim et stratégies gagnantes (fonction de Grundy en TD)
3. Jeux vidéo
   - Comment fait-on ?
   - Infographie
   - Gestion de l'IA
4. Jeux de stratégies (dérivés des jeux de société)
   - Réseaux

Casse-tête





- Problèmes solitaires (1 seul joueur)
  - une situation de départ
  - un objectif final
- Méthode : énumérer des solutions possibles
  - Tant qu'on a pas trouvé la solution :
    - On choisit d'une solution possible = qui vérifie toutes les contraintes spécifiées dans les règles du jeux
    - Si on arrive à une contradiction (les contraintes ne sont plus vérifiées), on retourne au dernier point de choix et on fait un nouveau choix…
    - Si on a exploré tous les choix, le problème n'a pas de solution

Que peut faire l'informatique ?

- Créer un langage ou il est facile de poser des contraintes (doit ressembler au langage naturel)

- Trouver une méthode de résolution des problèmes qui mimique la technique humaine
  - Modéliser les contraintes (règles du jeux)
  - Automatiser la « réflexion »

Contraintes ?

- Une contrainte est une **relation logique** (une propriété qui doit être vérifiée) entre différentes inconnues, **appelées variables**, chacune prenant ses valeurs dans un ensemble donné, **appelé domaine**.
- Une contrainte restreint les valeurs que peuvent prendre simultanément les variables.

Ex : "x + 3*y = 12" restreint les valeurs que l'on peut affecter simultanément aux variables x et y.

Définition d'un CSP

- Un CSP **(Problème de Satisfaction de Contraintes)**

- Ex :
  1. X = {A,B,C,D}
  2. D(A) = D(B) = D(C) = D(D) = {0,1} ou {A,B,C,D}∈ {0,1}
  3. C = { A ≠ B, C ≠ D, A+C < B }

Solution d'un CSP (1)

- Soit (X,D,C) un CSP. Sa résolution consiste à **affecter des valeurs aux variables, de telle sorte que toutes les contraintes soient satisfaites**.

- **Affectation (ou substitution)** = fait d'instancier certaines variables par des valeurs (évidemment prises dans les domaines des variables).

Ex : A_1={(B,0),(C,1)} est l'affectation qui instancie B à 0 et C à 1.

- Une affectation est dite **totale** si elle instancie toutes les variables du problème ; elle est dite **partielle** si elle n'en instancie qu'une partie.
- Une affectation A **viole une contrainte Ck** si toutes les variables de Ck sont instanciées dans A, et si la relation définie par Ck n'est pas vérifiée pour les valeurs des variables de Ck définies dans A.

Ex : l'affectation partielle A_2 = {(A,0),(B,0)} viole la contrainte A≠B ; en revanche, elle ne viole pas les deux autres contraintes dans la mesure où certaines de leurs variables (C,D) ne sont pas instanciées dans A_2.

Solution d'un CSP (2)

- Une affectation (totale ou partielle) est **consistante** si elle ne viole aucune contrainte, et **inconsistante** si elle viole une ou plusieurs contraintes.

Ex :

Une **solution** est une **affectation totale consistante**, c'est-à-dire **une valuation de toutes les variables du problème qui ne viole aucune contrainte.**

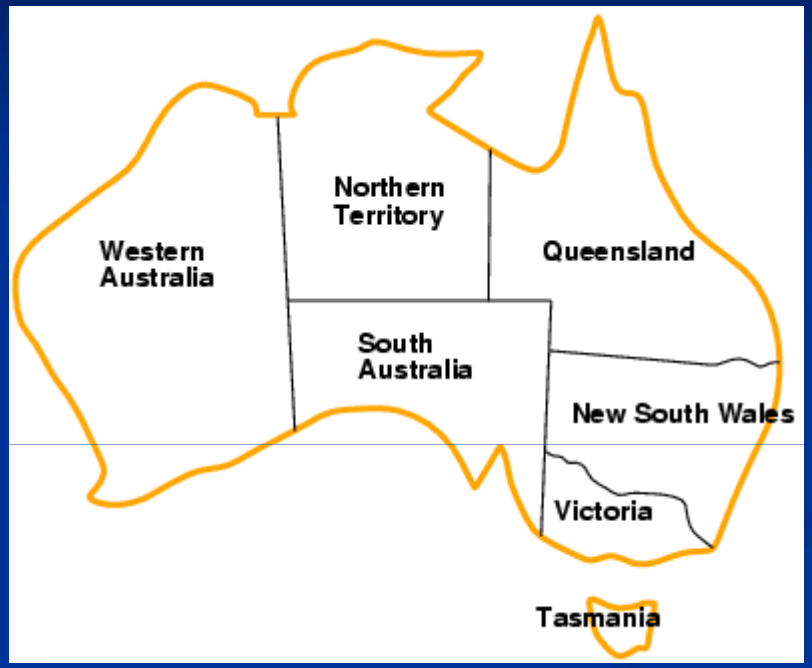
Ex :

Ce que vous ne verrez pas aujourd'hui… (RV en Master)

- Vérifier qu'une contrainte est « satisfaisable » est difficile !

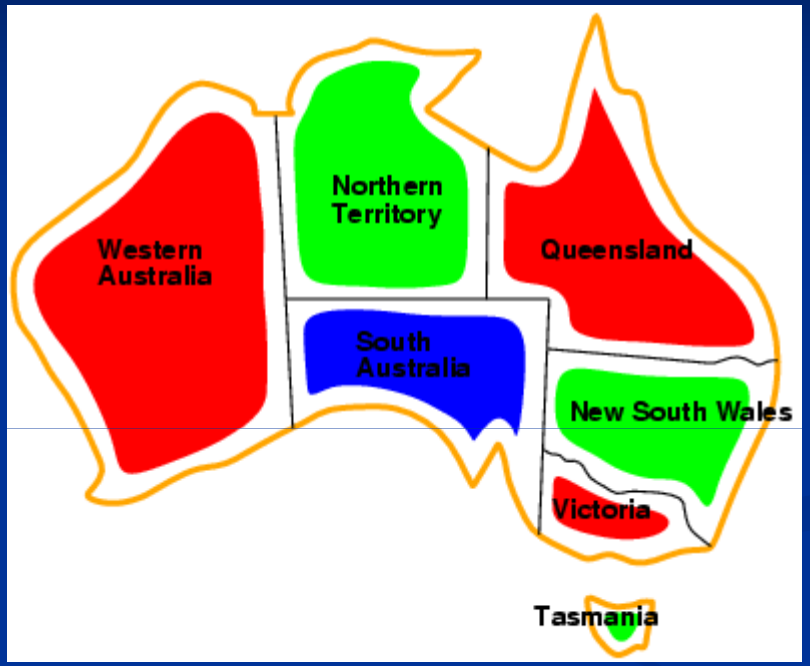
- Difficulté du problème dépend du type des contraintes (linéaires ou non) et des variables (entiers, réels,…)…

Exemple : Coloration de cartes (1)



- Variables *WA, NT, Q, NSW, V, SA, T*
- Domaine D_i = {rouge, vert, bleu}
- Contraintes: Les régions adjacentes doivent avoir des couleurs différentes ex: WA ≠ NT ou, (WA, NT) ∈ {(rouge, vert), (rouge, bleu), etc.}

Exemple : Coloration de cartes (2)



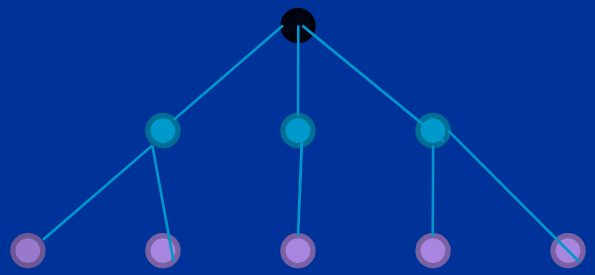
Solution: une affectation qui satisfait toutes les contraintes:

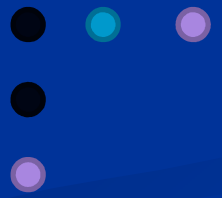
Exemple: Cryptarithmétique

```
  T W O
+ T W O
-------
F O U R
```

Arbres de recherche (détaillé dans un prochain cours)

- Arbre = graphe particulier qui contient une racine, des nœuds et des feuilles
  - Ex :





Nœuds
Racine
Feuilles

- Arbre de recherche = arbre dont les nœuds sont étiquetés par des affectations (partielles) du problème. La racine contient l'affectation vide, les feuilles contiennent des affectations totales.

Recherche de solutions (naïve)

- États (nœuds) : les valeurs assignées jusqu'à maintenant
  - État initial (racine) : l'affectation vide {}
  - Fonction successeurs (enfants) : affecter une valeur à une variable non affectée qui ne créée pas de conflit
  - Test de but (feuilles) : l'affectation est complète

- Commentaires :
  - Même formulation pour tous les CSP
  - Toutes les solutions à *n* variables apparaissent à la profondeur *n* dans un arbre de recherche –> recherche en profondeur d'abord

Exploration avec retour arrière

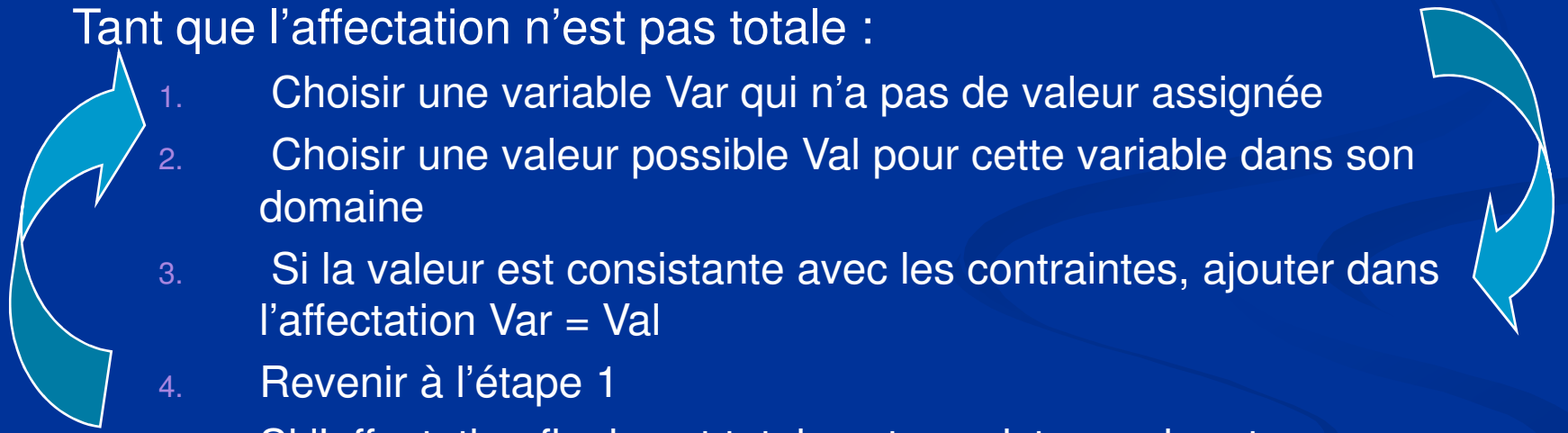
- L'assignation des variables est commutative
  - Ex: [*WA* = *rouge* suivi de *NT* = *vert*] est la même chose que [*NT* = *vert* suivi de *WA* = *rouge*]
- À chaque nœud de l'arbre, on a seulement besoin de considérer l'affectation d'une seule variable.
  - *var* = valeur et il y a |valeurs|$^{|variables|}$ feuilles
- La recherche en profondeur d'abord pour les CSP avec l'assignation d'une seule variable à chaque tour est appelé exploration avec retour arrière.
- Le retour arrière s'effectue lorsqu'il n'y a plus d'affectations possibles.
- Méthode très efficace qui peut résoudre de gros problème combinatoires (par ex : le problème des n reines vu en TD, peut se résoudre avec n = 25).

Principe de l'algorithme

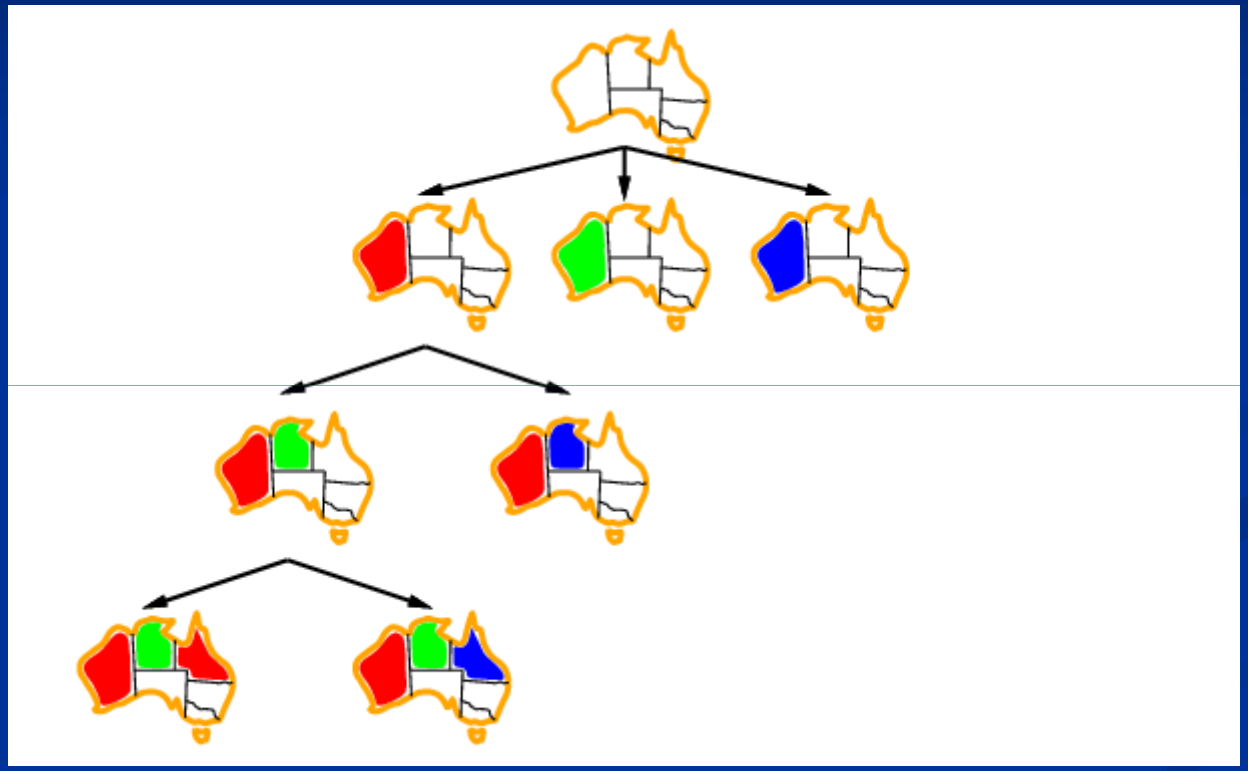
- But : rendre **affectation totale et consistante**
- Au début (initialisation) : une affectation vide {}
- Principe :

Tant que l'affectation n'est pas totale :

1. Choisir une variable Var qui n'a pas de valeur assignée
2. Choisir une valeur possible Val pour cette variable dans son domaine
3. Si la valeur est consistante avec les contraintes, ajouter dans l'affectation Var = Val
4. Revenir à l'étape 1
5. Si l'affectation finale est totale est consistance, la retourner en résultat
   - Sinon, enlever la valeur Var = Val dans l'affectation



Exemple retour arrière



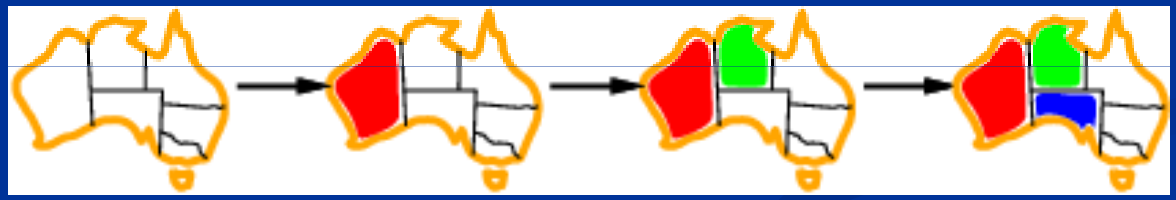
Améliorer l'efficacité de l'algo

- Quelle variable devrait être assignée au prochain niveau ?
- Dans quel ordre les valeurs devraient être essayées ?
- Quelles sont les implications des affectations de la variable courante sur les autres variables non affectées ?
- Lorsqu'un chemin échoue, est-ce que l'exploration pourrait éviter de reproduire cet échec dans les chemins suivants ?

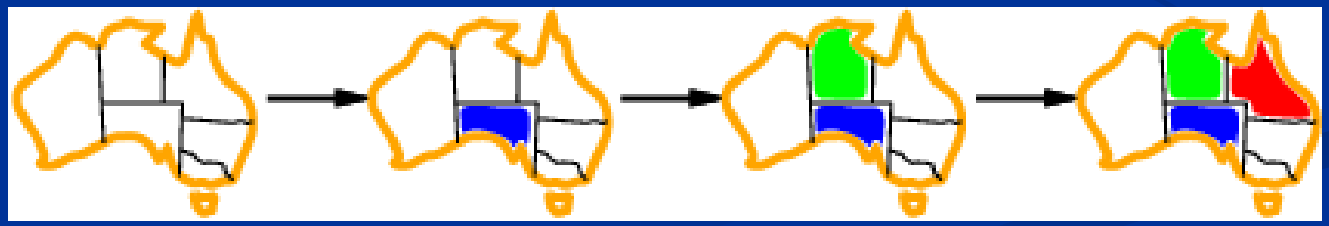
Ordre des variables

Choix des variables:

- Heuristique (=algo qui permet de trouver une « bonne » solution à un « prix » (ex temps) acceptable) du nombre minimum de valeurs restantes
  - Choisir la variable ayant le moins de valeurs possibles.



- Heuristique du degré
  - Choisir la variable présente dans le plus de contraintes.

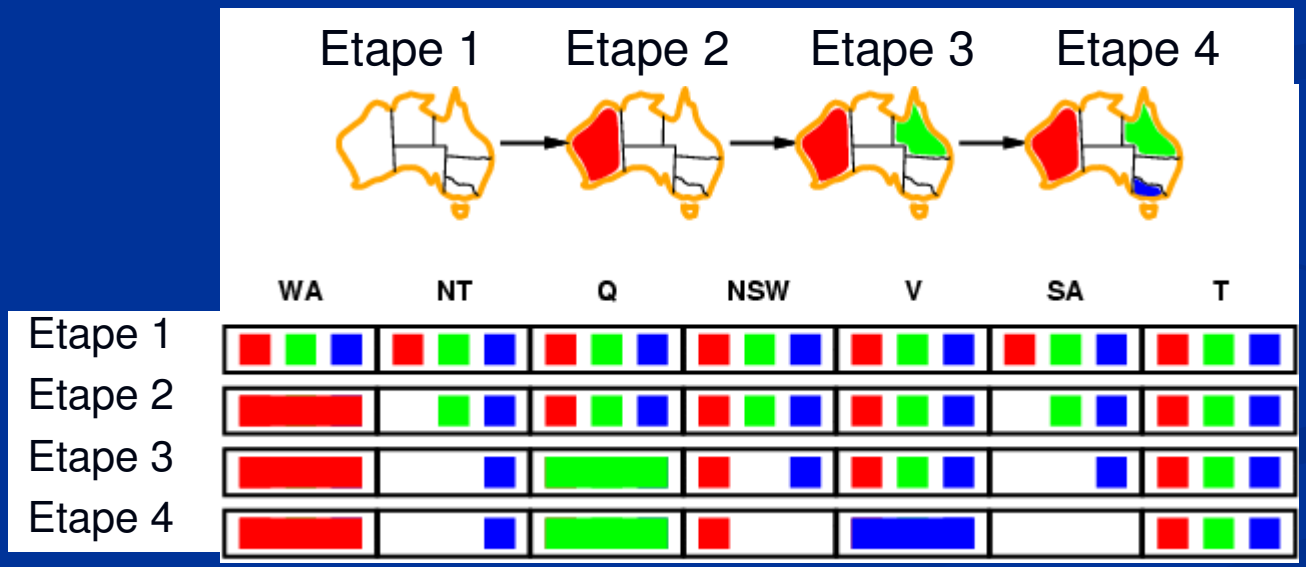


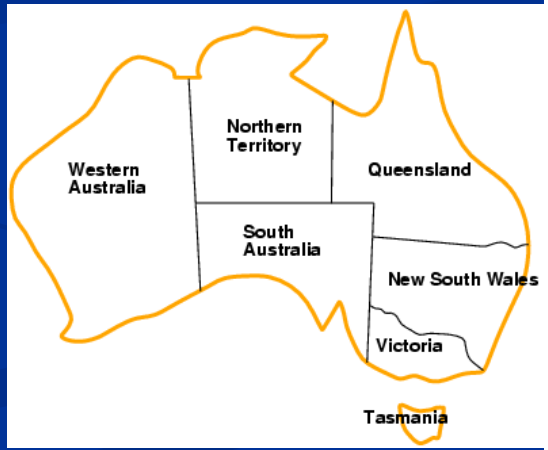
Ordre des valeurs

- La valeur la moins contraignante
  - Choisir la valeur qui élimine le moins de choix pour les variables voisines.

Forward checking

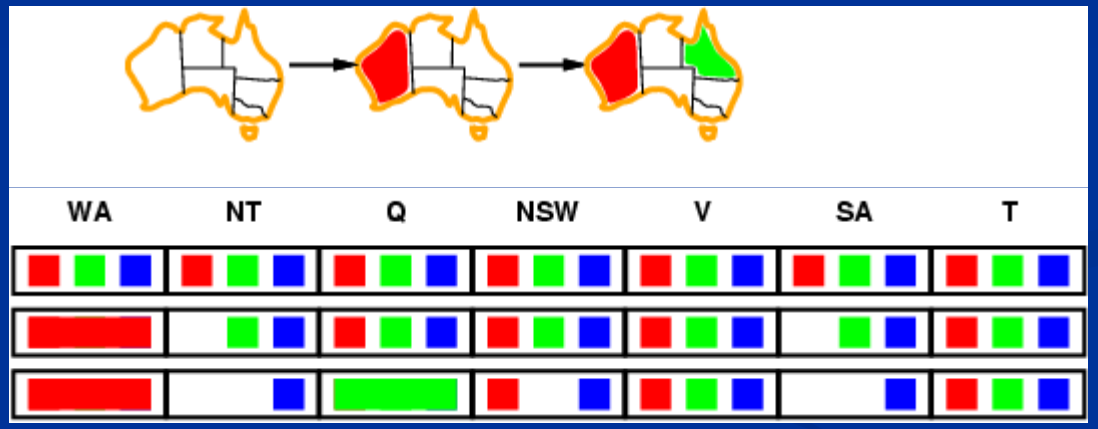
- Maintient en tout temps, toutes les valeurs possibles pour chacune des variables.





Propagation de contraintes

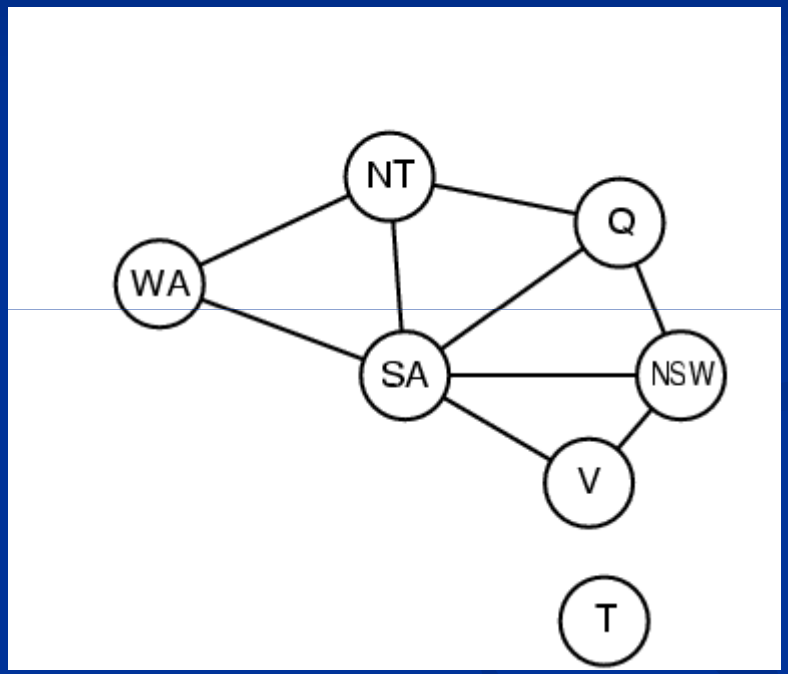
- Le « *forward checking* » ne permet pas de détecter tous les problèmes.



- NT et SA ne peuvent pas tous les deux être bleus!
- La propagation de contraintes renforce rapidement les contraintes locales.

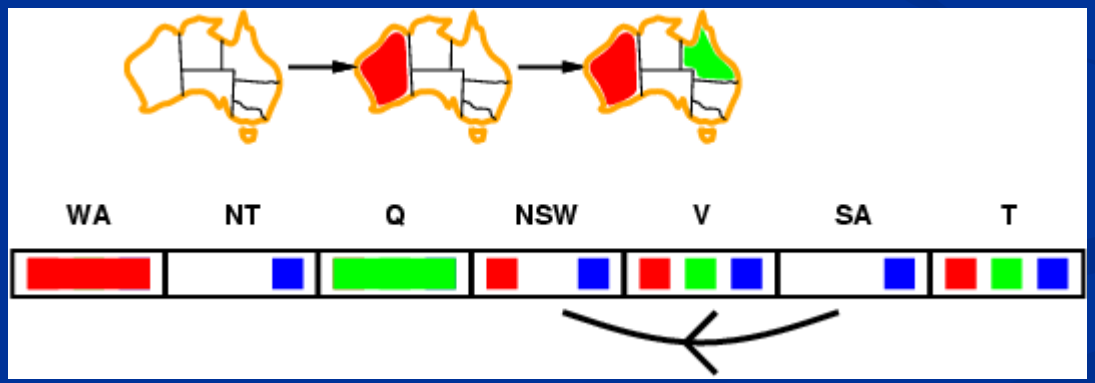
Représentation des contraintes

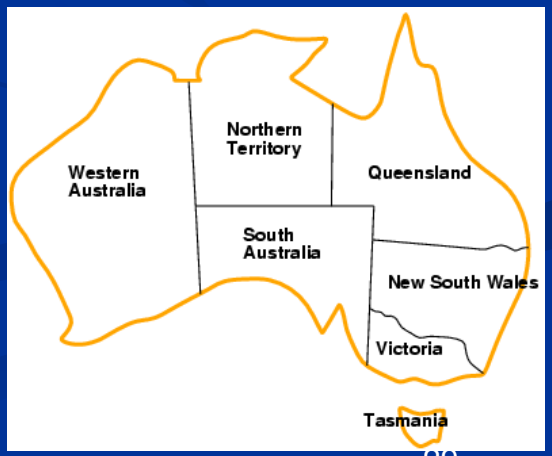
- CSP où toutes les contraintes sont reliées à deux variables.
- Peuvent être représentés sous forme de graphe.
- Certains algorithmes généraux de résolution de CSP utilisent la structure de graphe pour accélérer la recherche.
  - Ex: Tasmanie est un sous problème indépendant.



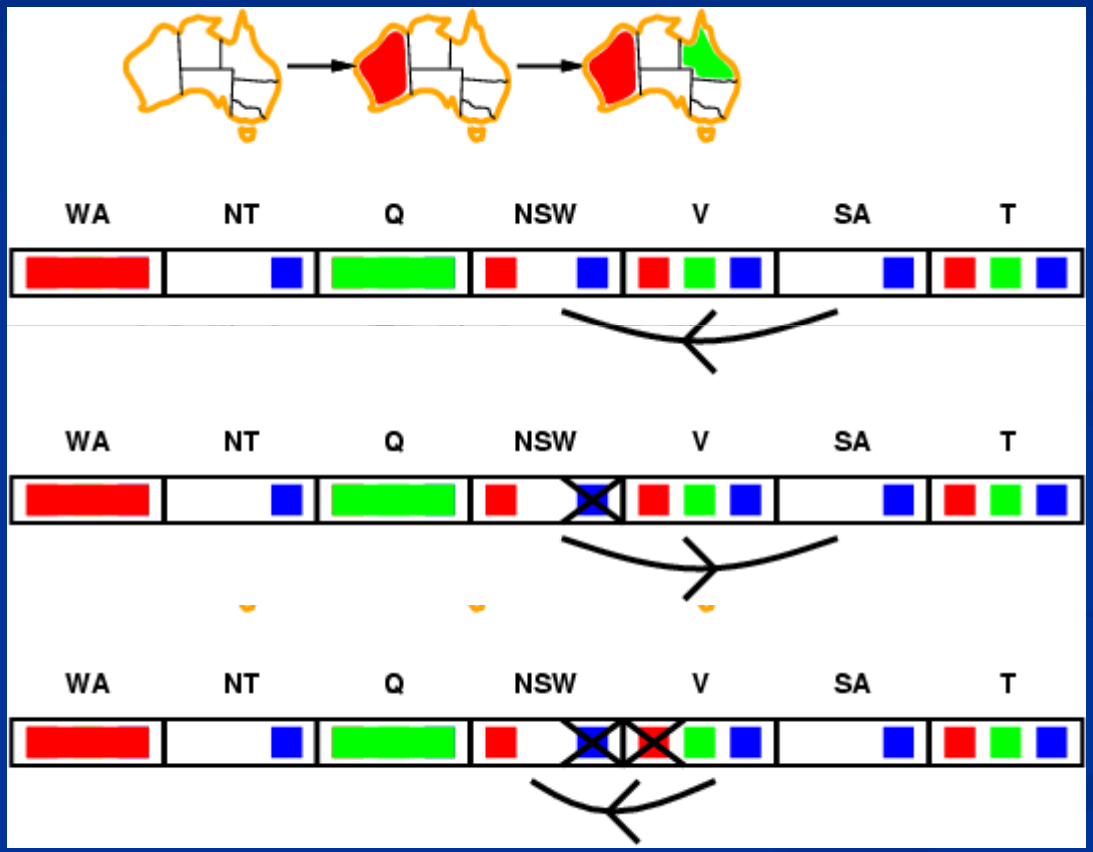
Cohérence des arcs (1)

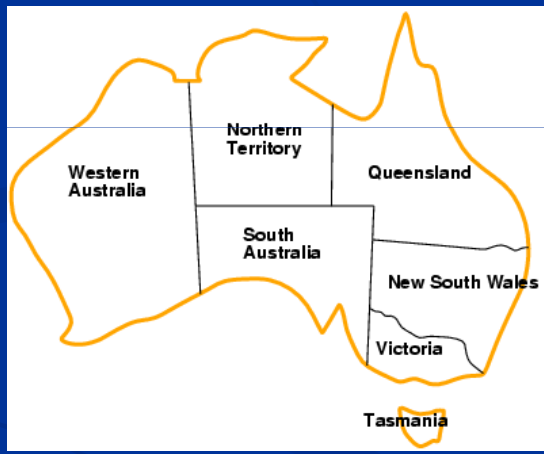
- Forme la plus simple de propagation : faire en sorte que chaque arc soit cohérent (permet de trouver des branches de l'arbre menant à un échec plus vite que le « forward checking »)
- But : réduire le domaine des variables
- Un arc entre *X* et *Y* est cohérent si pour toutes les valeurs *x* de *X*, il y a au moins une valeur possible *y* de *Y*.





Cohérence des arcs (2)





Si X perd une valeur, les voisins de X doivent être revérifiés

Jeux « classiques »

- Différence : jeux à plusieurs (souvent deux) joueurs.
- Pas de « solution unique».
- Jouer contre un «ordinateur », est-ce possible ?
- Un peu d'histoire…

Culture générale
(pour en savoir plus)

- Musée de l'histoire de l'ordinateur **(en Anglais),** partie sur l'informatique est les échecs en ligne :
- http://www.computerhistory.org/chess/interact/ (présentation interactive)

http://www.computerhistory.org/chess/main.php?sec=thm-42b86ebff1a81&sel=thm-42b86ebff1a81 (histoire en texte)

Les progrès de l'IA pour les jeux "classiques"

- Construire un programme informatique capable de jouer au échecs aussi bien que le meilleur des humains était l'un des premiers "grand défi" de la recherche en Intelligence Artificielle (IA)…
- C'était supposé être un problème simple….mais il s'est avéré beaucoup plus compliqué qu'on ne l'imaginait ☺

Alex Bernstein
(IBM, late 1950s)

Défi sous-estimé…

"Within ten years a computer will be the world's chess champion, unless the rules bar it from competition."

Herbert Simon 1957 (inventeur de l'algo minmax, Carnegie Mellon, USA)

Le maître David Levy domine le monde des échecs pendant plus de 20 ans

La recette des vainqueurs

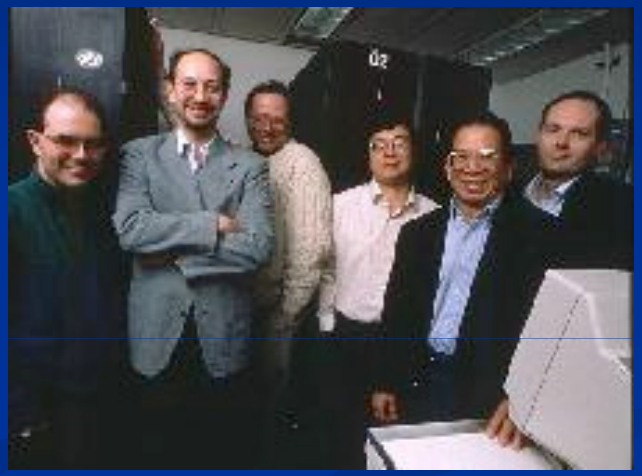
- Algorithme MiniMax avec recherche heuristique Alpha-beta (vu en cours et TD sur les arbres)
  - Paraît simpliste pour un humain…
- Du matériel rapide !!!

- Mise en place qu'à partir du milieu des années 80…

Echecs (1997)

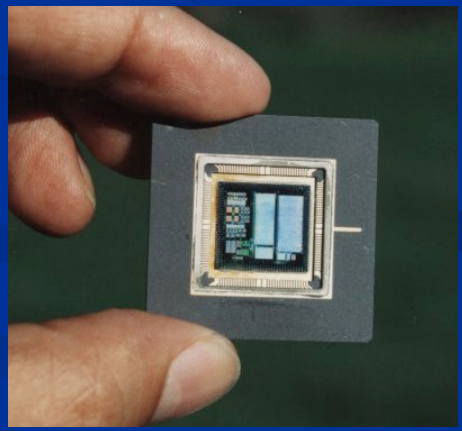
Garry Kasparov.
Champion du monde d'échecs (1997).

IBM :
Feng Hsu, Murray Campbell, et Joe Hoane



Deep Blue.

Synergie entre l'homme et la machine



Dames (2007)

Marion Tinsley. Homme…où machine?
A perdu seulement 3 matchs en 40 ans !
(jusqu'en 1994)

En 1994 : l'équipe *Chinook* (http://www.cs.ualberta.ca/~chinook/) gagne le championat du monde de dames Homme-Machine

En 2007 : le jeu de dame est entièrement résolu : si le noir joue en premier, il ne peut faire que match nul contre un opposant parfait

Othello (1997)

*Logistello (*Michael Buro, Alberta, CA) utilise des méthodes de recherche et des algorithmes d'apprentissage et bas Takeshi Murakami, le champion du monde.

Takeshi Murakami. Le résultat du match (6-0) confirme que les hommes n'ont aucune chance contre les machines…

Backgammon (1998)

Malcom Davis (un des meilleur joueur du monde pendant 20 ans)→ le backgammon n'est pas un jeu de chance ☺

Gerald Tesauro (IBM) créé un programme qui joue "aussi bien" qu'un humain expert : *TD-Gammon* (utilise des techniques d'apprentissage par renforcement)

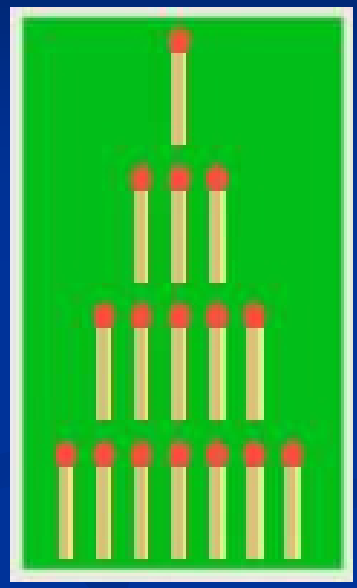
Un vrai défi : le poker

- Etat de l'art? Programmes "optimaux"
  - Basés sur la théorie des jeux (voir cours Initiation aux sciences sociales = dilemme du prisonnier)
  - Joue bien mais ne gagne pas beaucoup (d'argent)
- Comment faire d'un programme un joueur de niveau international ? (Patrick Bruel)
  - L'ordinateur doit modéliser ses opposants et biaiser les probabilités de miser pour exploiter les tendances des adversaires.

Jeux de Nim

- Une variante vue en TD sur les arbres (jeu de Marienbad)

- Caractéristiques :
  - jeux de duel à somme nulle (deux joueurs, un vainqueur et un perdant, pas d'égalité possible). L
  - le nombre d'éléments en jeux diminue au fur et à mesure de la partie
  - Une stratégie optimale de gain existe, basée sur la reconnaissance de positions intermédiaires gagnantes.



Stratégies gagnantes (voir TD)

- Ex de règle : Au départ il y a n allumettes, chaque joueur prend à tour de rôle 1, 2 ou 3 allumettes, et le joueur qui prend la dernière allumette a perdu. Celui qui laisse une seule allumette a donc gagné.

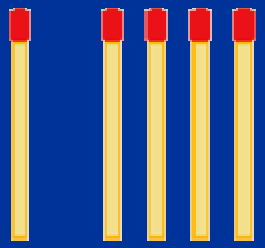
- Jeux à deux
- Seulement 12 « états » possibles : il reste 12, 11, 10… ou 1 allumette.
- Il existe une stratégie gagnante ! (une manière de jouer pour être sûre de gagner)

Déroulement du jeu (1)

- Imaginons que je joue contre vous, si la situation 1 est la suivante et que c'est à vous de jouer :

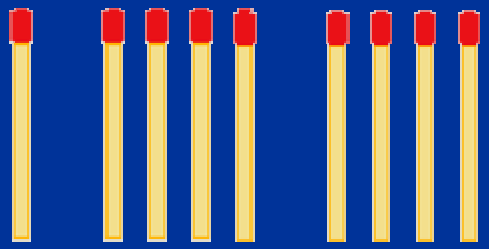
Déroulement du jeu (2)

- Si nous sommes maintenant dans cette situation 2 (n = 5) et que c'est encore à vous de jouer :



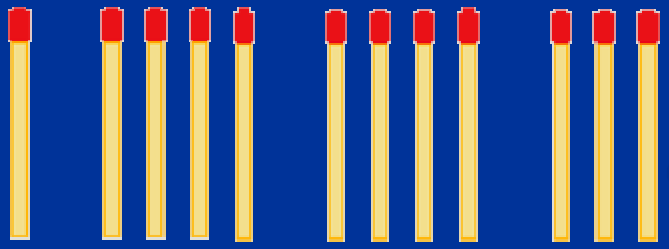
Déroulement du jeu (3)

- Maintenant, si nous sommes dans cette situation 3 avec encore 4 allumettes de plus (n= 9) et que c'est encore à vous de jouer :



Déroulement du jeu (4)

- Il est possible de continuer comme ça en ajoutant autant de fois que l'on veut 4 allumettes. Ainsi, par exemple, dans un jeu d'allumettes avec n = 12 :



Que fait l'ordinateur ?

- Dans les jeux de Nim, quelle que soit la stratégie gagnante utilisée, elle peut être implémentée dans un ordinateur.
- Les algorithmes qui représentent une stratégie gagnante utilisent la théorie des graphes et plus particulièrement la définition du noyau d'un graphe (voir TD pour la suite)

Les temps changent…

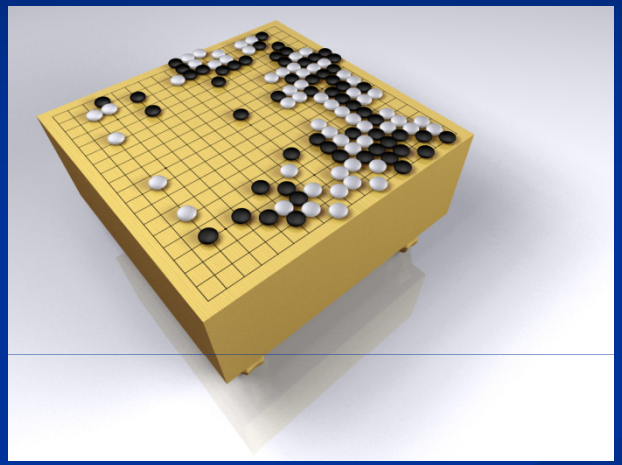
- Les jeux de société et de cartes “classiques” n’intéressent plus les étudiants de nos jours…

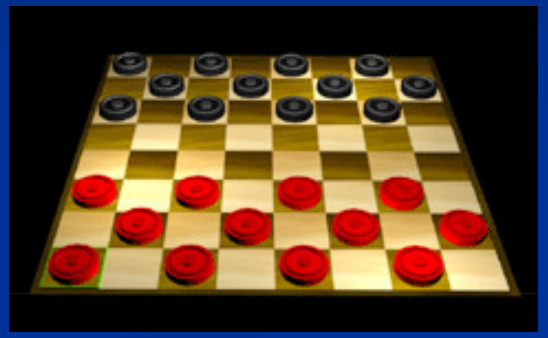
(= moins de recherche donc moins d’avancées)

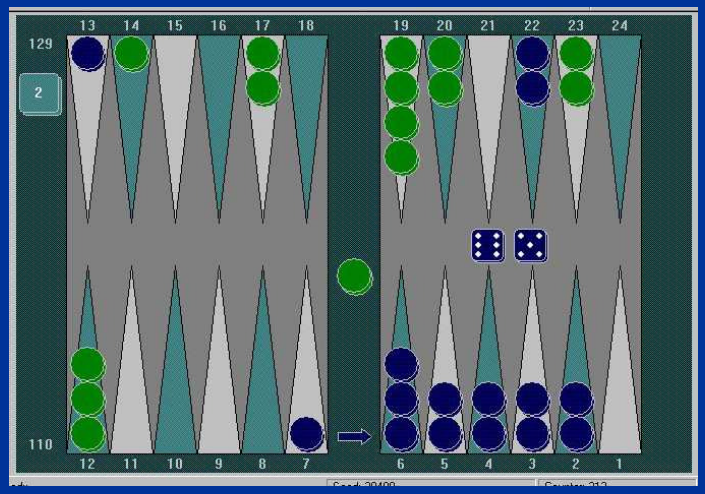
- L’industrie des jeux « pèse » 30 billions de dollars
- De plus en plus besoin de nouvelles technologies (IA, matériel, graphismes …) dans les nouveaux produits.

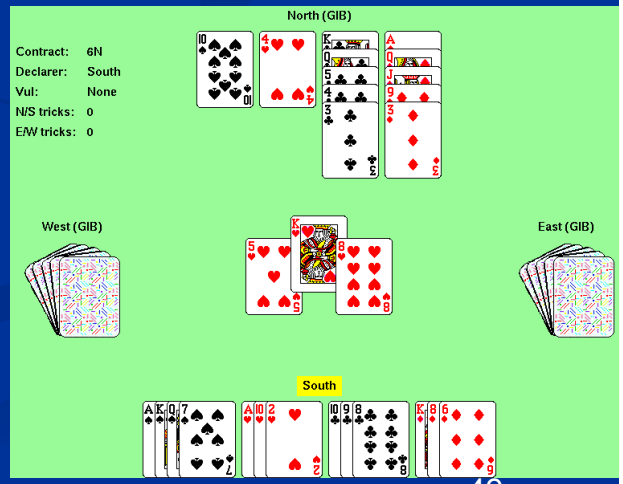
De plus en plus d’opportunités de faire un travail industriel pertinent…..et de s’amuser !!

Au lieu de ça…






13 14 15 16 17 18 19 20 21 22 23 24
129
2
110
12 11 10 9 8 7 6 5 4 3 2 1

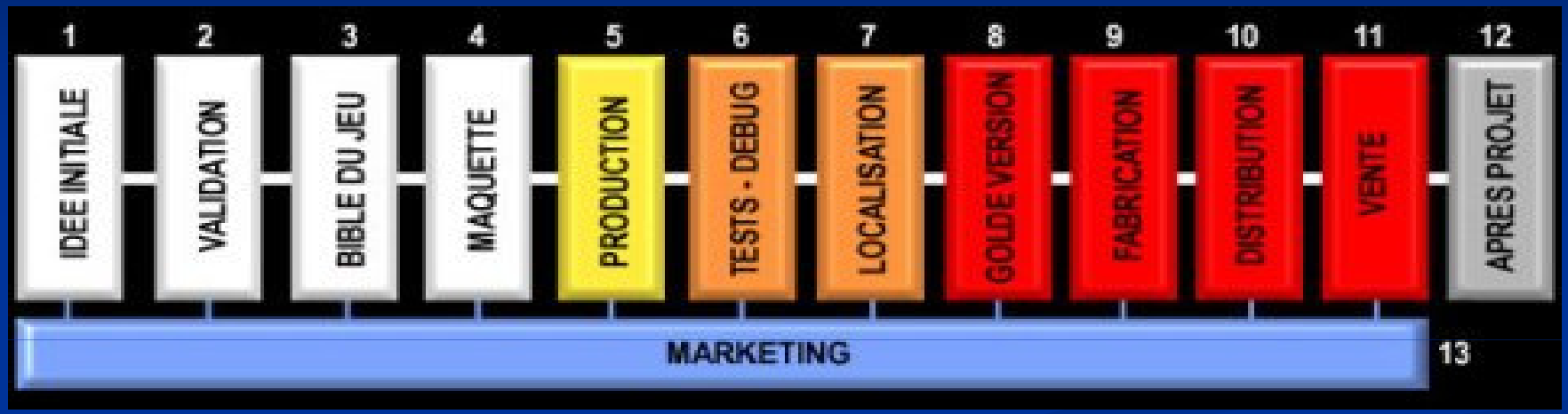

North (GIB)
Contract: 6N
Declarer: South
Vul: None
N/S tricks: 0
E/W tricks: 0
West (GIB)
East (GIB)
South


Unlimited Adventures...
... faites ça!
Neverwinter Nights
BioWare Corp
Atari

L'informatique et les jeux

2 ème partie

Comment faire un jeu vidéo ?



Chronologie d'un projet

- Validation du concept (« proof of concept ») (2-4 mois)
  - Convaincre l'éditeur pour démarrer une pré-production
- Pré-production (6 mois-1 an)
  - Recherche gameplay (scénario, interaction, joueur, jouabilité hors graphique), charte graphique
  - Recherche technologique (gameplay/IA, graphisme, animation)
  - Création de la chaine de production (logiciels utilisés pour construire le jeu)
  - FPD (First Playable Demo)

Chronologie d'un projet

- Production (1-3 ans)
  - Développement du gameplay
  - Création des graphismes, animations
  - Finalisation et amélioration de la chaine de production
  - Présentation à l'éditeur / presse
  - Période de test / debug
- Soumissions X360, PS3, etc... (1 mois)
  - Le jeu respecte-t-il les règles de la console ?
  - Master candidate

Quelques chiffres

- Nouveau projets (Xbox 360, PS3) : Assassin's Creed 2, Bioshock 2, Gran Turismo 5, etc...
  - 2-3 ans de travail
  - 100-300 personnes
    - Graphistes (modèleur, animateur, textureur) 60
    - Level designer/Game designer (création des niveaux de jeu, du gameplay) 50
    - Sound designer 10
    - Programmeur (graphique, physique, gameplay, outils) 50
    - Management 20
    - Testeur 80
  - 15-20 millions d'euros (production uniquement)

Programmeurs de jeu vidéo

- Programmeur graphique
  - Rendu 3D des environnements, calcul des ombres, éclairage, optimisation du rendu
  - Algèbre linéaire, algorithmique, optimisation
- Programmeur physique
  - Simulation de modèles réels (voitures, personnages, objet dynamique, fluides)
  - Physique, algorithmique, optimisation
- Programmeur animation
  - Animation de personnage (marche, course, combats, escalade)
  - Animation par clé, cinématique inverse, optimisation

Programmeurs de jeu vidéo

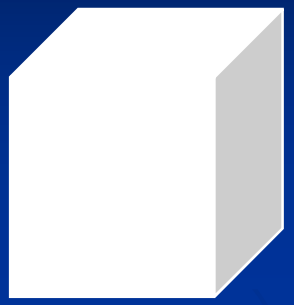
- Programmeur IA / gameplay
  - IA des personnage non joueur (adversaires, déplacement des PNJ), gameplay divers (missions, système économique)
  - Algorithmique des graphes, machine à état, logique, algorithmique
- Programmeur outils
  - Création du logiciel d'édition (gameplay, graphique, animation), gestion des ressources du jeu (+500 000 modèles graphiques)
  - Algorithmique, génie logiciel

Les contraintes techniques

- Graphismes réalistes, animations réalistes, IA efficace, physique, ambiance sonore, réseau… INTERACTION !
  - En temps réel (30-60 images par seconde)
  - Calculer une image en moins de 20 ms !
- Consoles avec des limites
  - Ne sont pas des super-calculateurs (mais quand même très efficace)
  - Limite mémoire vive
- Les DVD sont limités
  - De plus en plus de données à mettre sur le disque (monde ouvert, diversité des personnages, des animations, etc...)

Techniques d'infographie

- Modélisation
  - Modèles paramétriques
  - Modèles hiérarchiques
  - Déformations du modèle
- Animation
  - Cinématique inverse
  - Résolution des équations différentielles
  - Particules, masse-ressort…
- Rendu (temps-réel)
  - Modèles complexes de matériaux
  - Ombres en temps-réel
  - Affichage temps-réel : niveaux de détail, etc.



MODELISATION

INSPIRÉ DE LA PRÉSENTATION DE NICOLAS HOLZSCHUCH (INRIA RHONE-ALPES)

Modélisation d'objets 3D

- Primitives graphiques = volumes simples
  - Cube (parallélépipède)
  - Sphère
  - Cône…
- Objet complexe = composition de primitives graphiques (section, intersection…)
- Représentation vectorielle des points
  - Points attachés aux primitives graphiques
  - Sommets, centres, données volumiques…

Transformations géométriques

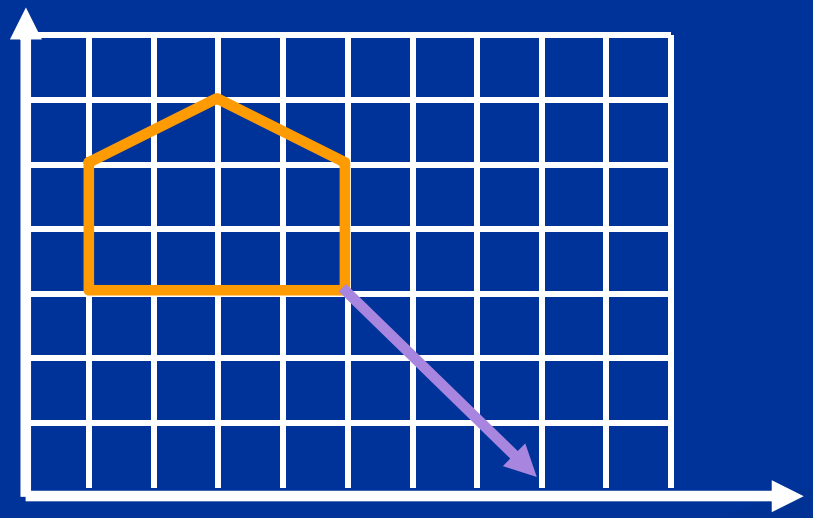
- Positionner les objets par rapport à un référentiel global (monde) ou local (objet)
  - Translation, rotation, changement d'échelle…
  - Projections (pour le rendu):
    - Perspective, parallèle…
  - Notation unifiée ? (on veut décrire toutes les transformations et leur composition par un même outil mathématique)

Transformations en 2D

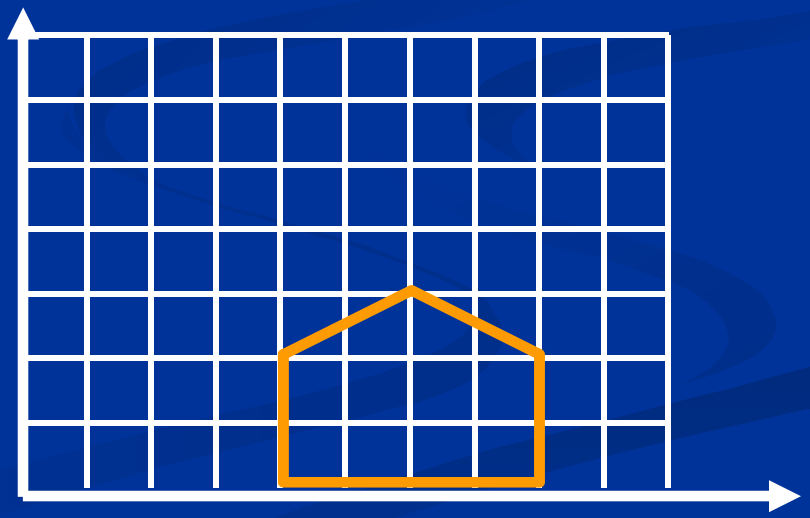
- On commence en 2 dimensions (2D)
  - Plus facile à représenter
- Chaque point est transformé:
  - $x' = f(x,y)$
  - $y' = g(x,y)$
- Comment représenter la transformation ?

Translation

- Modification simple :

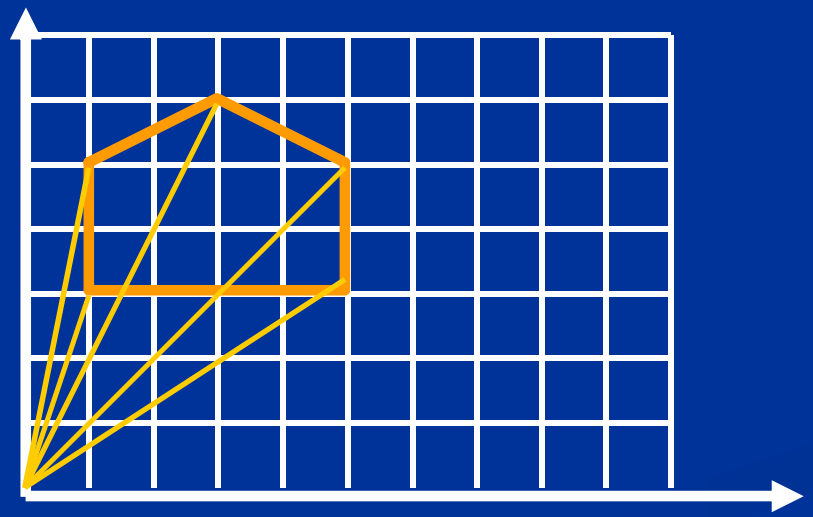


Avant

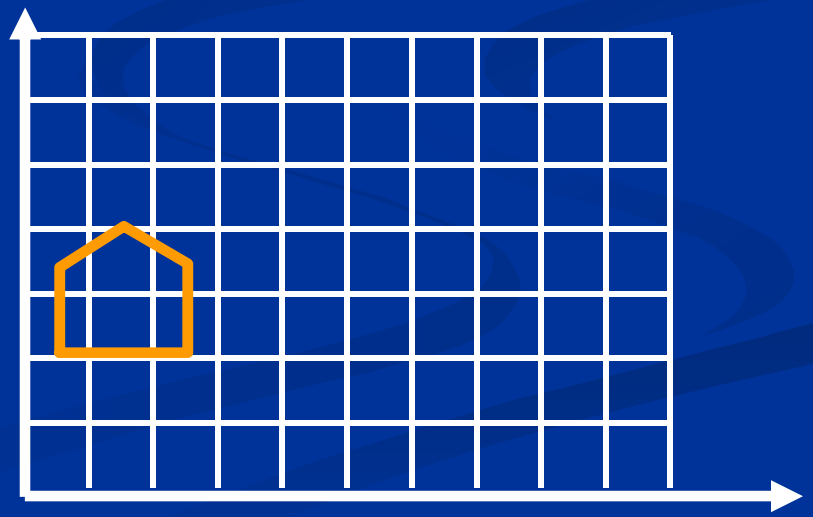


Après

Changement d'échelle



Avant



Après

Notation matricielle

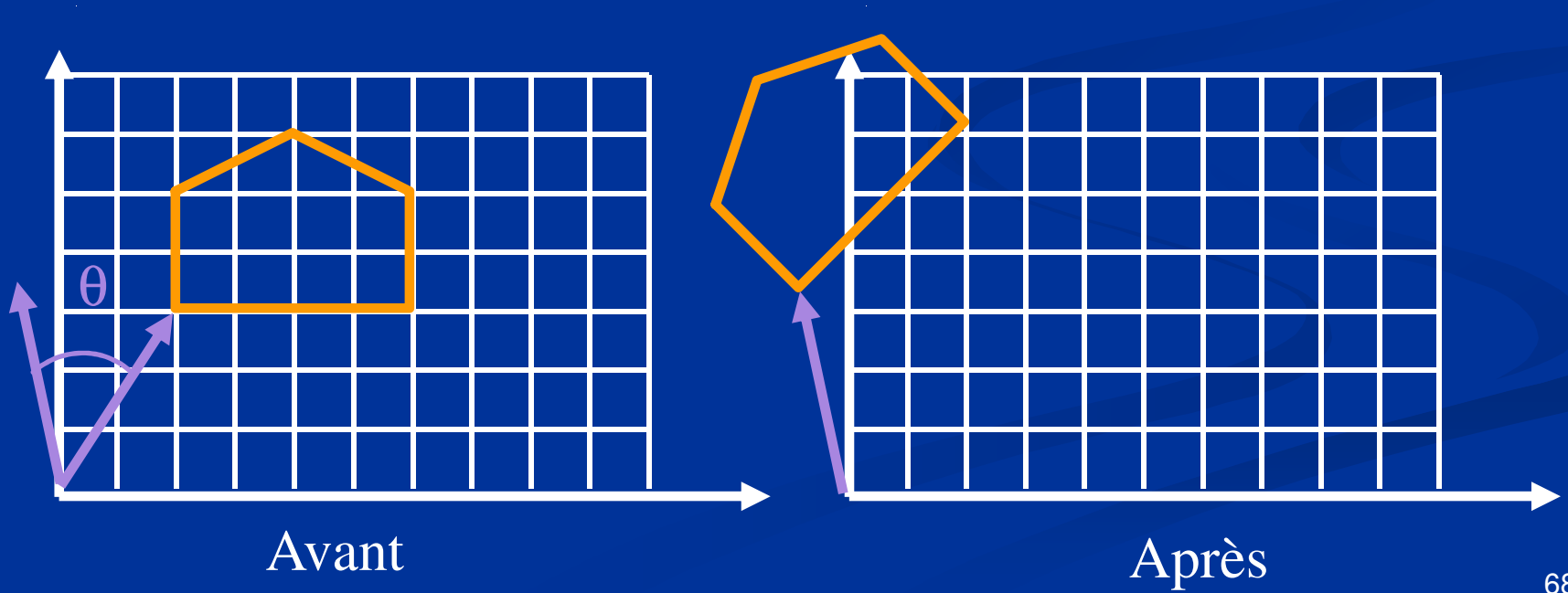
- On utilise une multiplication matricielle pour représenter un changement d'échelle

$$P' = \mathbf{S}P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

- Rotation en 2D :



Notation matricielle

- Représentation de la rotation : multiplication matricielle

$$P' = \mathbf{R}P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Unification des transformations

- Pour l'instant :
  - Notation simple, concise
  - Mais pas vraiment unifiée
    - Addition ou bien multiplication
    - Comment faire pour concaténer plusieurs transformations ?
- On veut une notation unique
  - Qui permette de noter aussi les combinaisons de transformations
  - Comment faire ?

Coordonnées homogènes

- Outil géométrique très puissant :
  - Utilisé partout en Infographie (Vision, Synthèse)
  - *cf.* aussi géométrie projective
- On ajoute une troisième coordonnée, *w*
- Un point 2D est représenté par un vecteur à 3 coordonnées :

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Coordonnées homogènes

- Deux points sont égaux si et seulement si :
  - $x'/w' = x/w$ et $y'/w' = y/w$
- *w=0:* points « à l'infini »
  - Très utile pour les projections, et pour certaines splines (= fonction définie par morceaux par des polynômes)

Translations 2D en coordonnées homogènes

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = x + wt_x \\ y' = y + wt_y \\ w' = w \end{cases}$$

$$\begin{cases} \frac{x'}{w'} = \frac{x}{w} + t_x \\ \frac{y'}{w'} = \frac{y}{w} + t_y \end{cases}$$

Changement d'échelle 2D en coordonnées homogènes

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = s_x x \\ y' = s_y y \\ w' = w \end{cases}$$

$$\begin{cases} \frac{x'}{w'} = s_x \frac{x}{w} \\ \frac{y'}{w'} = s_y \frac{y}{w} \end{cases}$$

Rotation 2D
en coordonnées homogènes

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = \cos\theta x - \sin\theta y \\ y' = \sin\theta x + \cos\theta y \\ w' = w \end{cases}$$

$$\begin{cases} \frac{x'}{w'} = \cos\theta \frac{x}{w} - \sin\theta \frac{y}{w} \\ \frac{y'}{w'} = \sin\theta \frac{x}{w} + \cos\theta \frac{y}{w} \end{cases}$$

Composition des transformations 2D

- Toutes les transformations 2D peuvent être exprimées comme des matrices (3x3) en coordonnées homogènes
  - Notation très générale
- Pour composer, il suffit de multiplier les matrices :
  - Ex : composition d'une rotation et d'une translation

$$\mathbf{M} = \mathbf{RT}$$

Rotation autour d'un point Q

- Rotation autour d'un point Q:
  - Translater Q à l'origine ($\mathbf{T}_Q$),
  - Rotation autour de l'origine ($\mathbf{R}_\Theta$)
  - Translater en retour vers Q ($\mathbf{-T}_Q$).

$$P' = (-T_Q) R_\Theta T_Q \; P$$

Et en 3 dimensions ?

- Même principe
- On introduit une quatrième coordonnée, *w*
  - Deux vecteurs sont égaux si :
    x/w = x'/w', *y/w = y'/w'* et *z/w=z'/w'*
- Toutes les transformations sont des matrices 4x4 de coordonnées homogènes

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translations en 3D

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{cases} x' = x + wt_x \\ y' = y + wt_y \\ z' = z + wt_z \\ w' = w \end{cases}$$

Changement d'échelle en 3D

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{cases} x' = s_x x \\ y' = s_y y \\ z' = s_z z \\ w' = w \end{cases}$$

Rotations en 3D

- Rotation : un axe unitaire et un angle
- La matrice dépend de l'axe et de l'angle
- Expression directe possible, en partant de l'axe et de l'angle, et quelques produits vectoriels
  - Passage par les quaternions pour homogénéiser les compositions
- (Souvent fait par une librairie graphique)

Rotation autour de l'axe des *X*

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L'axe des X n'est pas modifié

Vérification: une rotation de π/2 devrait changer *y* en *z*, et *z* en -*y*

$$R_x(\tfrac{\pi}{2}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation autour de l'axe des *Y*

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L'axe des Y n'est pas modifié

Vérification: une rotation de π/2 devrait changer *z* en *x*, et *x* en -*z*

$$R_y(\tfrac{\pi}{2}) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation autour de l'axe des *Z*

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L'axe des Z n'est pas modifié

Vérification: une rotation de π/2 devrait changer *x* en *y*, et *y* en -*x*

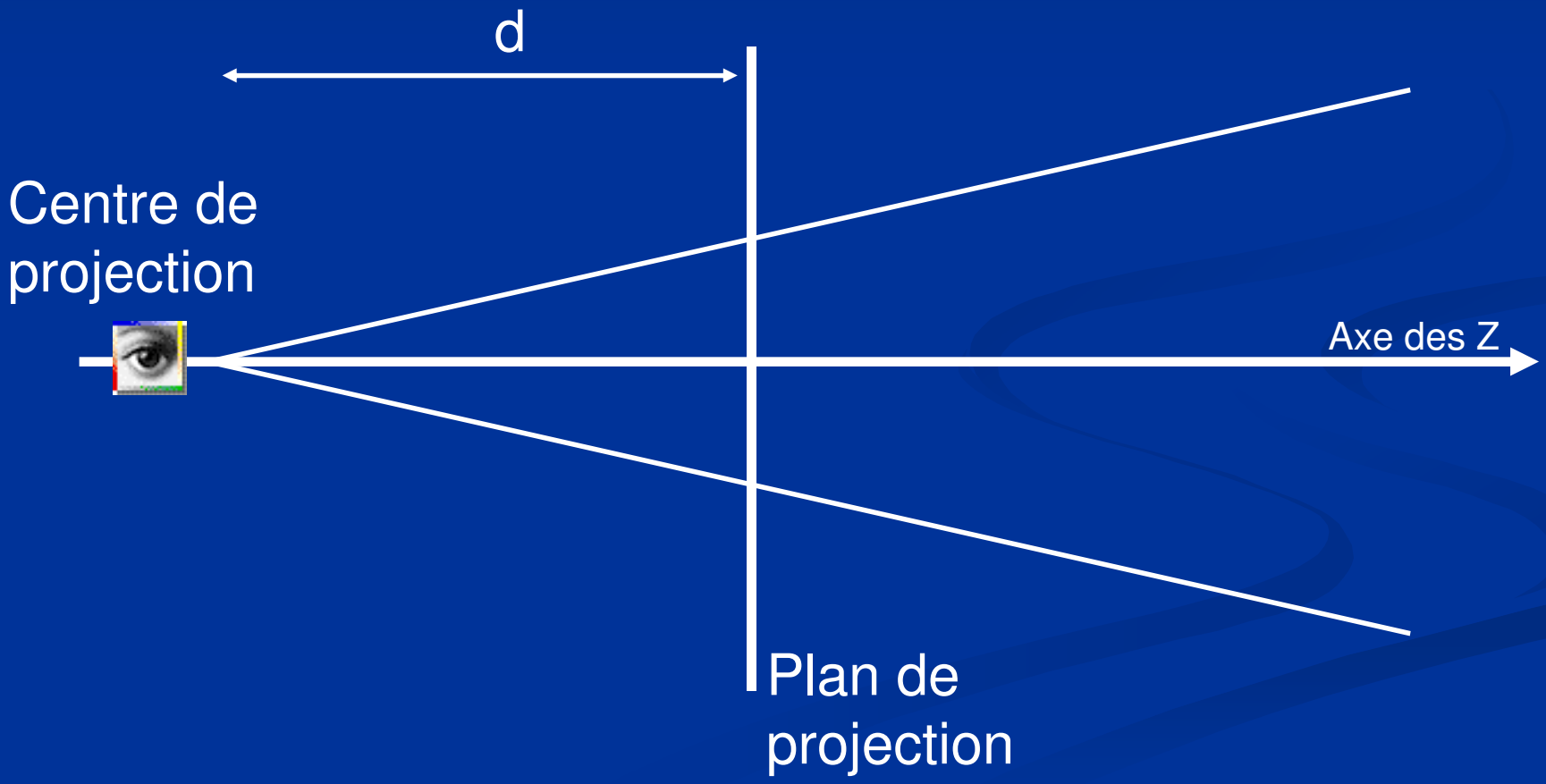
$$R_z(\tfrac{\pi}{2}) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations 3D

- Toute transformation 3D s'exprime comme combinaison de translations, rotations, changement d'échelle
  - Et donc comme une matrice en coordonnées homogènes
- (souvent fourni par une librairie graphique)
  - glTranslatef(x, y, z);
  - glRotatef(angle, x, y, z);
  - glScalef(x, y, z);

Perspective

Hors champ = pas de rendu



Projection perspective

- Pour les objets qui ne sont pas hors-champ
- Projection sur le plan *z=0*, avec le centre de projection placé à *z=-d*:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

Projection perspective (suite)

- Coord. homogènes essentielles pour perspective
- La rétrécissement des objets utilise *w*

$$w' = \frac{z}{d}$$

$$\frac{x'}{w'} = \frac{d * x}{z}$$

- Impossible sans coordonnées homogènes

Modélisation paramétrique

- Dit aussi *modélisation procédurale*
  - Modèle fait par un programme
  - Fondamental en infographie
- Paramètres de position
- Paramètres de forme
  - Forme des parties de l'objet
  - Position relative des parties de l'objet

Exemple : bonhomme de neige

Exemple : bonhomme de neige

- Trois sphères empilées
- Paramètres :
  - Écrasement
  - Inclinaison
  - Pour chaque sphère

ANIMATION

INSPIRÉ DE LA PRÉSENTATION DE NICOLAS HOLZSCHUCH (INRIA RHONE-ALPES)

Animation

- Différentes techniques d'animation :
  - Entrée par l'utilisateur
  - Key-frame interpolation
  - *Motion capture*
  - Cinématique
  - Cinématique inverse
  - Dynamique
- Choix en fonction des contraintes
  - Scenario, réalisme…
  - Mélange de méthodes

Animation manuelle

- Action directe de l'animateur sur les paramètres de position, orientation, etc.
  - Ex : on déplace un objet à la souris
- Mouvement fourni par la souris
  - À partir des coordonnées à l'écran on détermine une action dans la scène 3D (x_0, y_0) et (x_t, y_t)
- Relation entre action de la main et du modèle
  - Perception logique
- Difficile d'agir sur modèle complexe
  - Quel partie de l'objet ?

Animation par « key-frame interpolation »

- Technique issue des dessins animés
- Animateur fournit données en entrée
  - Positions-clefs, vitesse… à temps t_i
- Interpolation entre points de contrôle
  - Utilise des courbes de Bézier 2D (splines)
- Possibilité d'introduire des contraintes mathématiques (pour respecter la physique)
- OK pour films
- Difficile pour les jeux (interaction et réalisme du mouvement)

Animation par *Motion Capture*

Animation par *Motion Capture*

- On enregistre la position 3D de capteurs placés aux articulations d'un acteur
  - Mouvement très réaliste
- On applique le mouvement sur le squelette 3D d'un personnage virtuel
  - Difficulté de l'adaptation (conservation du réalisme ?)
- Nécessite des grosses bases de données de mouvement (ou matériel coûteux)

Animation par cinématique directe

- Vitesse donnée en entrée
  - Programme calcule la position
- Utile pour des objets simples soumis à la pesanteur, trajectoires simples
- Contrôle complet de l'objet
  - Pratique pour suivre le *scenario*
- …mais *besoin* d'un contrôle complet de l'objet
  - Difficile pour l'animateur

Animation par cinématique inverse

- Objets complexes (pas seulement soumis à la pesanteur)
  - Ex : bras articulé = chaîne articulée
- Animation d'une partie de l'objet (ex : main)
- Calcul des positions du reste de l'objet
- Simple pour animateur/joueur (guidé par le but)
- Problème complexe
  - Non-linéaire, pas d'unicité, pas de continuité…

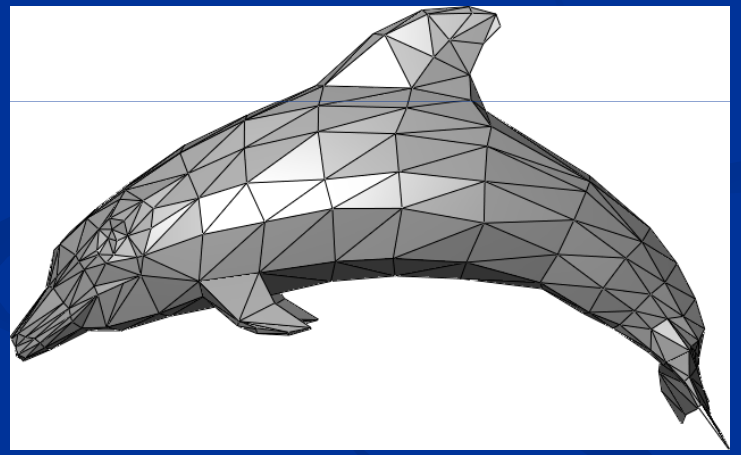
Animation basée sur la dynamique

- Lois de la dynamique, appliquées au modèle d'animation (s'applique conjointement aux techniques vues précédemment)
- Trajectoires réalistes
  - Si modèle réaliste
- Complexité pour déterminer et introduire les contraintes mathématiques (Ex : motion capture)
- Utile pour particules, objets secondaires…

RENDU
(= CALCUL DE L'IMAGE À AFFICHER)

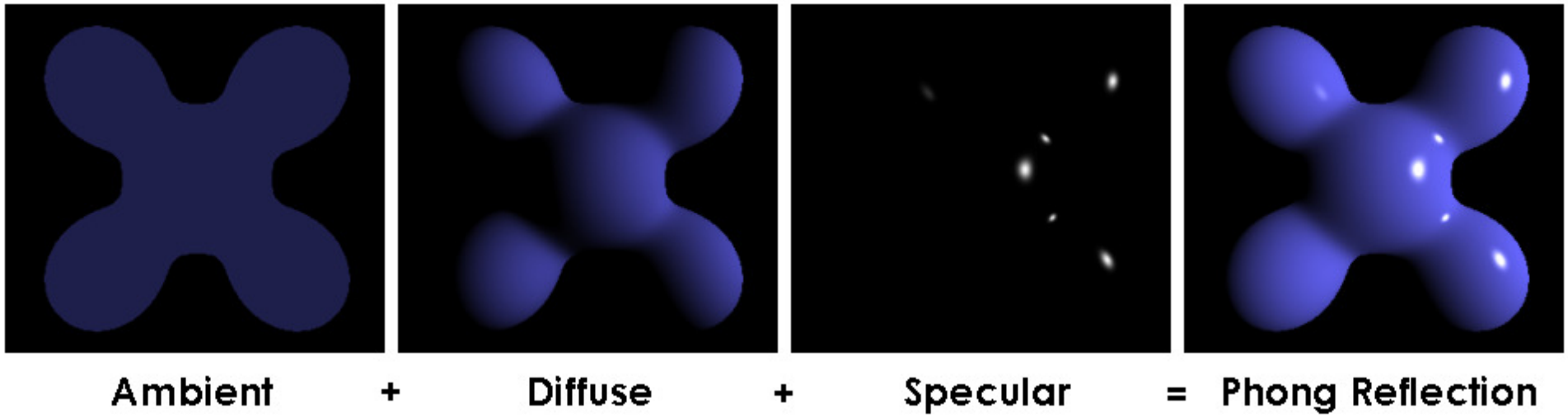
Conditions du rendu 3D temps réel

- Les graphistes construisent des modèles 3D (logiciel 3DSMax, Maya, ZBrush) = ensemble de triangles, connectés entre eux + texturés (matériaux)
  - Affichage des modèles en temps réel
  - Transformation du modèle (translation, rotation, changement d'échelle)
  - Affichage des matériaux : couleur, texture
  - Eclairage du modèle (gestion sources de lumière, ombres)
  - Déformation du modèle (animation)
  - Déplacement de la caméra dans l'environnent
  - Optimisation du rendu (on n'affiche que ce qu'on voit) beau et rapide !
  - Faire semblant, pas de la synthèse d'image photoréaliste



Ex : Eclairage 3D

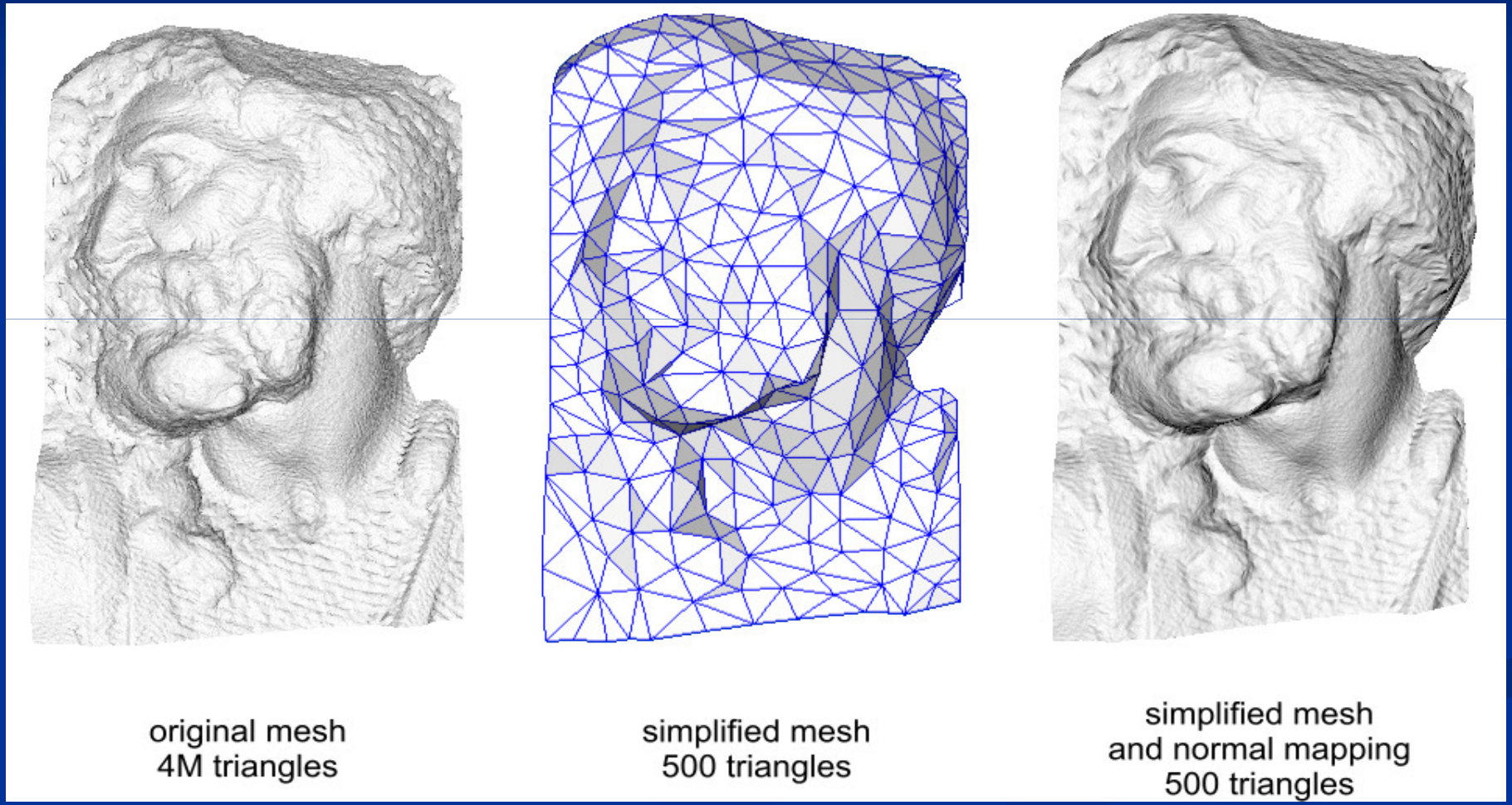
- Modèle 3D non éclairé, on ajoute
  - Ambient (constant sur l'environement)
  - Diffuse (éclairage mat, dépend de la source de lumière)
  - Spéculaire (éclairage miroir, dépend de la source de lumière et de l'observateur)
- Exemple modèle d'éclairage de Phong



Ex : Optimisation 3D

- Modèle 3D avec peu de triangle
  - Moins de place en mémoire
  - Plus rapide sur la carte graphique
- Comment conserver la qualité graphique tout en affichant rapidement ?
  - Donner l'illusion !
- Exemple : « normal map »
  - L'oeil créé le volume grâce aux ombres
  - L'éclairage est modifié pour créer des ombres et créer un faux volume
  - Permet de réduire le nombre de triangle tout en conservant la qualité

Ex : Optimisation 3D



Images de synthèse

- Processus à plusieurs étapes
  - Modélisation (+ artiste)
  - Animation
    - Keyframe, cinématique, contraintes, dynamique
    - Aspects réaliste
    - Contraintes de l'histoire
  - Rendu
    - Affichage
    - Aspect réaliste
    - Contraintes de temps

Films d'animation

- Story-board
- Modèle géométrique
  - Dessin, sculpture, scanner 3D
- Animation :
  - Mouvements à grande échelle (ex : foule)
  - Mouvements précis
- Rendu :
  - Rendu rapide pour vérification (tps réel)
    - ex pour vérifier le réalisme des « key frames »
  - Rendu complet :
    - 90 mn par image, 25 img/sec, 1h30 de film = 202500 h de calcul

Ce qui rend un jeu meilleur…

- Jusqu'à récemment, la qualité graphique (images, animation,…) était l'atout le plus important d'un produit
  - La vitesse des avancées technologiques s'est réduite
  - Les cartes graphiques ont nivelé les champs d'actions possibles
- La qualité graphique attire les gens vers un jeu mais le "game-play" les pousse à continuer à jouer (et à acheter les suites ☺)
- L'industrie du jeu vidéo reconnait que le développement de l'IA est crucial pour les jeux de nouvelle génération

Les jeux et l'IA

- Il y a 15 ans, IA dans le jeux commerciaux était primitive
- Les algo de recherche (type A*) on été "découverts" en 1996…
- Maintenant, la plupart des jeux ont un développeur en IA à temps complet, souvent deux
- Les ressources données à l'IA sont toujours disproportionnellement petites (comparé à la qualité graphique)

L'IA dans les jeux c'est…

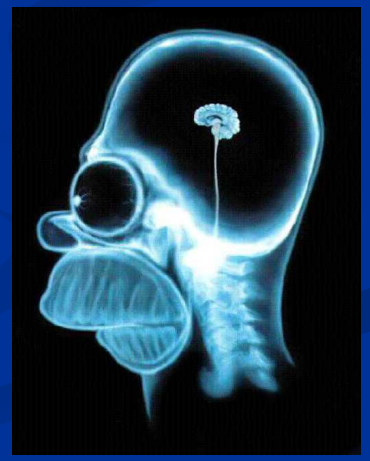
- Créer l'illusion d'un comportement "intelligent"
  - Malin mais pas trop (a-t-on envie de perdre tout le temps ???)
  - Imprévisible mais prenant des décisions rationnelles
  - Influençable émotionnellement
  - Communiquant ses émotions avec un "langage corporel"
  - Intégré à l'environnement de jeux

- Avec un monde de plus en plus réaliste, il devient de plus en plus dure d'induire une illusion…

- Les développeurs de jeux perdent cette bataille [Pieter Spronck : chercheur en IA pour le jeux]

Stupidité artificielle

- Les joueurs ont tendance à ce souvenir des choses “stupides” qui leur arrivent dans les jeux
  - Comportements répétitifs
  - Joueurs qui marchent à travers les murs
  - Absence d’apprentissage…



- C’est amusant…….un temps.

Etat de l'art ?

Radiant A.I. : enfin un système d'IA innovant ! (2006)

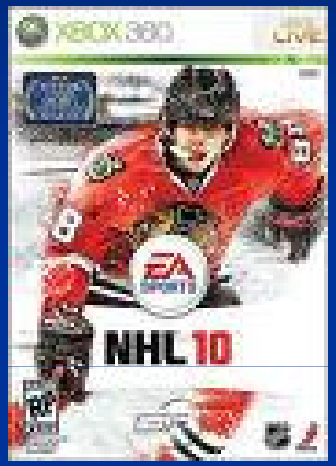
Personnages réalistes: le jeu met en scène plus de 1000 personnages non-joueurs plus vivants que jamais grâce à l'animation faciale et à la synchronisation des lèvres avec des dialogues complets. Les PNJ se mettent même à entamer des discussions spontanées entre eux !

Voir la démo pour s'en rendre compte…

L'informatique et les jeux

3 ème partie

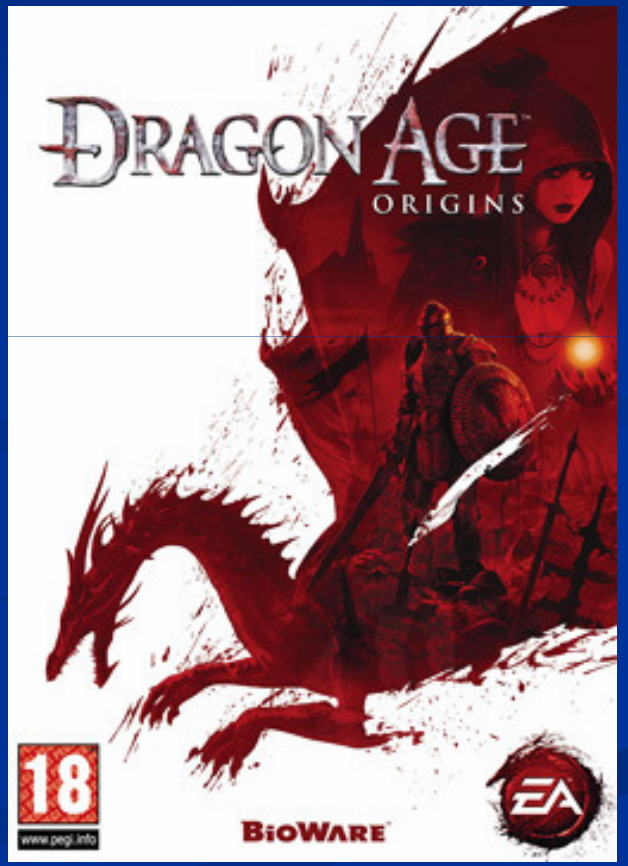
Jeux de sport (1)



Jeux de sport (2)

- Le “réalisme” est critique
  - Equipe – beaucoup de données
  - Joueur – observe, bouge et agit comme une personne
  - Action – utilisation de capture du mouvement
- Etat de l’art en IA ?
  - Systèmes à base de règles
  - *Nécessité de modéliser les opposants*
  - *Ajuster le jeu dynamiquement pour s’adapter à l’humain*

Jeux de rôle/ Action-aventure



Jeux de rôle

- Le scénario et l'intrigue
- Personnages autonomes
- Conversations
- Etat de l'art (en IA)
  - "Scripting" (dialogue pré-écrit)
  - Automates à états finis
  - Pas de génération dynamique du contenu
  - Pas d'apprentissage (essentiel pour créer un environnement immersif)

Jeux de simulation (1)

Jeux de simulation (2)

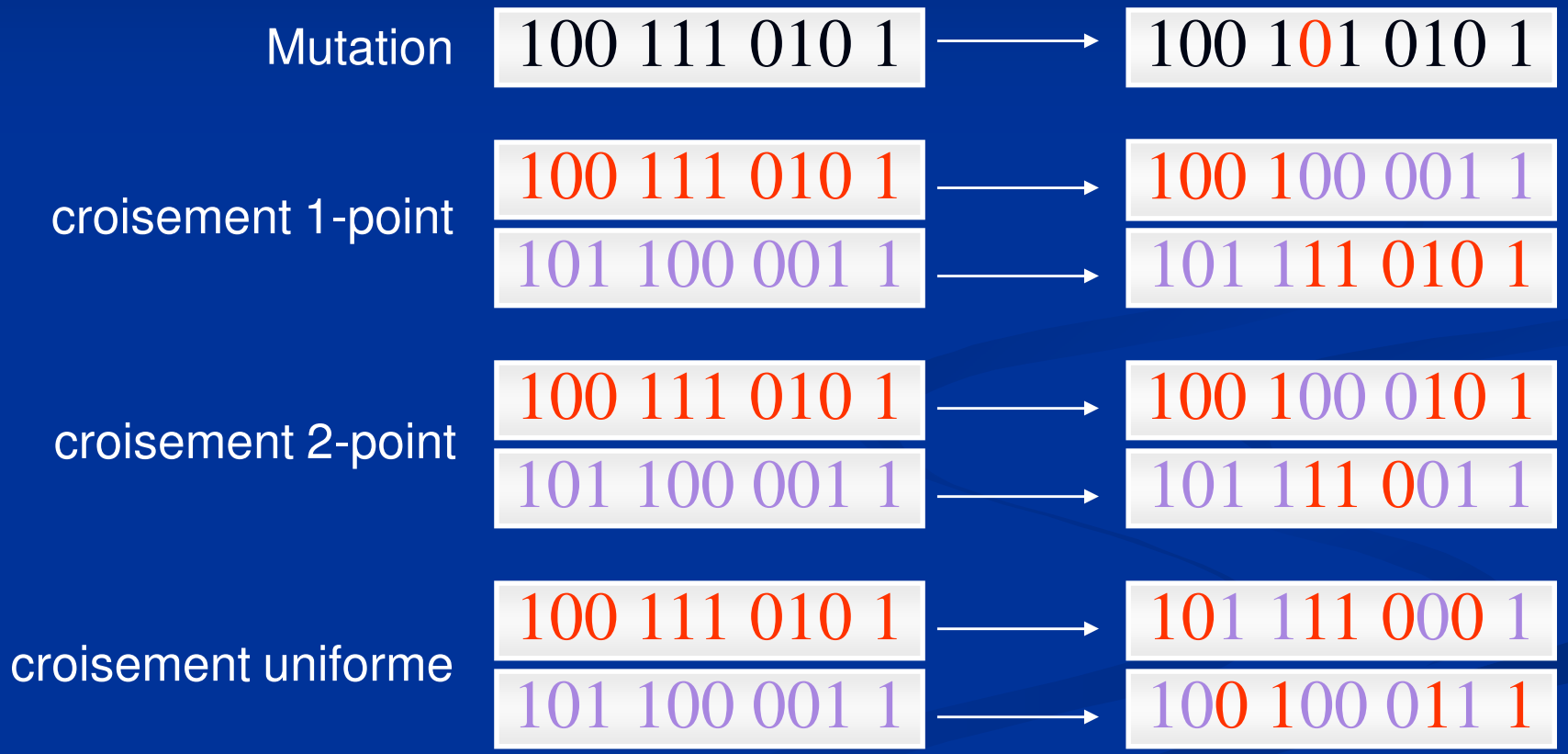
- “Terrain intelligent”
- Chaque objet connait son propre ensemble de comportements et réagit aux stimuli extérieurs
- Etat de l’art en IA?
  - Automates “flous” à états finis
  - Vie artificielle
  - Algorithmes d’apprentissage simples

Algorithmes évolutionnaires (~1960)

- Basés sur les principes d'évolution naturelle
  - Lamarck, Darwin, Wallace: évolution des espèces, survie du plus adapté ("fittest")
  - Mendel: la génétique fournit des mécanismes d'héritage
- Essentiellement : une procédure de recherche massivement parallèle
  - On commence avec une population d'individus aléatoire
  - On évolue graduellement vers de "meilleurs" individus en optimisant la fonction d'adaptation (fitness)
- Cas le plus connus des algorithmes évolutionnaires : algorithmes génétiques

Principe de l'algorithme

Exemples d'opérateurs (un code génétique fait de 0 et 1)



Vie Artificielle ?

- Vidéo (transparent suivant)
- But : développer un “poisson tueur” capable de manger le plus de nourriture en un temps constant

- Chaque poisson est un réseau de neurones (voir M1)
- Caractéristiques qui évoluent d’un génération à l’autre :
  - En fonction :
    - De la vitesse et de la direction du poisson lui-même
    - De la position relative de la nourriture la plus proche
  - Changer la vitesse de l’aile droite et la vitesse de l’aile gauche
- A chaque génération on garde les 20 meilleurs poissons (ceux qui mangent le plus) et on opère des mutations

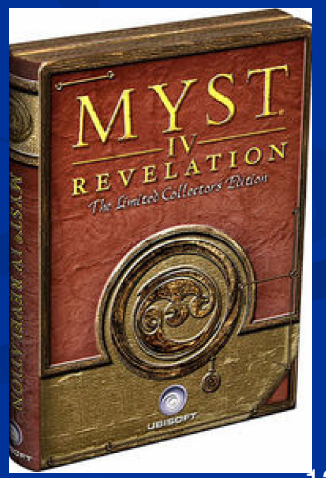
Stratégie temps-réel (1)

Stratégie temps-réel (2)

- Des milliers d'agents qui demande une réponse en temps réel
- Le jeu doit poser un vrai défi et demande des stratégies poussée et une réflexion rapide
- Etat de l'art en IA ?
  - Scripting (bout de programme qui sont déclenchés au fur et à mesure
  - Technique de planifications (voir en M1)
  - Besoin de modéliser l'opposant (pas fait !)
  - Adapter le jeu dynamiquement pour rivaliser au bon niveau avec le joueur (pas fait !)

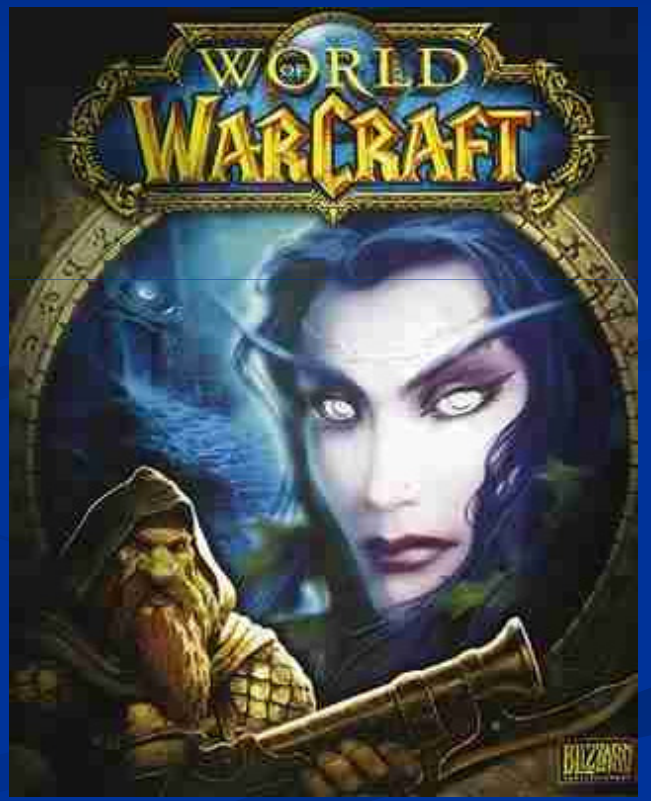
D’autres genres…

- *First-person shooter*
- *Action games*
- *Racing games*
- *Adventure games*
- *Children’s games*
- *Educational games*
- *Classic games*



World of Warcraft (WoW)

- Massively Multiplayer Online RPG (MMORPG)
- 11,5 million d'inscrits (fin 2008)
  - Population des Pays Bas….
- Plus de 1 Milliard d'euros de recette en 2006 (pour Vivendi games)
  - Surpasse tous les revenus rapportés par tous les jeux vidéo (hors console) en 2005
- Le succès vient du fait que les opposants dans le jeux sont VRAIMENT intéressants...



Quelle est la technique d'IA la plus utilisée dans les jeux?

- Les nombres aléatoires ☺
- Les utilisateurs ont tendance à identifier des comportements intelligents dans des séquences aléatoires…

- Idéal : nombres aléatoires contrôlés…

Goopy Gilscarbo (sims)

I have to confess, I've grown a bit attached to Goopy. Here's how it happened. The very first family I made was just a single mother and a toddler. I hoped to find a nice, unselfish man for her to marry. Instead she met Goopy. He moved in and she had a child with him, but his behavior towards her made me hate him. I planned an exciting, cruel death to give Goopy. I had everything in place, and then I clicked on his picture to find him and send him to his death. He had his son (who was a toddler) in his arms, and he was cuddling his child. :eek: That just melted my heart, and after that I couldn't bring myself to kill him. So Goopy lives on, and he's become a very good father.

Goopy Gilscarbo

L’avenir de l’IA dans les jeux ?

- Mettre à la poubelle votre matérielle de première classe
- Utiliser le moins possible de temps CPU et de mémoire
- Arriver à généraliser avec peu de données
- Garantir des performances bornées
- Donner une réponse temps-réel
- Créer du réalisme
- Créer une IA au même niveau que l’humain…

Attention…

“La programmation d’IA pour les jeux ne doit pas être confondue avec la recherche académique en IA : la programmation des jeux n’utilise que très peu la recherche en IA. Même si ces deux activités ont des points communs, elles sont généralement considérées comme deux disciplines distinctes.” wikipedia

>95% de la recherche académique appliquée aux jeux n’a aucun intérêt pour l’industrie…

Et les jeux de sociétés dans tout ça….

De plus en plus d'adaptations sur ordinateur !

Par exemple : http://www.brettspielwelt.de/?nation=fr

Différences…

- Développement très simple
  - Nombre d'actions déterminées : règles claires et « finies »
  - Pas ou très peu de 3D
- Challenge :
  - Pas passionnant de jouer contre une IA… (beaucoup moins « ludique » tout seul qu'un jeu vidéo classique)
  - Faire jouer les hommes ensemble dans tous les coins de la planète (comme WoW) → réseaux !!

Réseaux

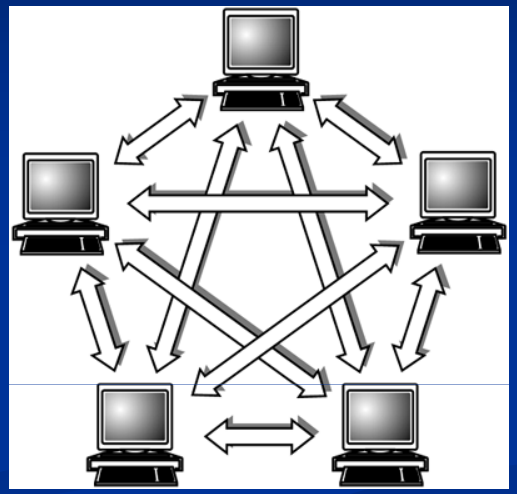
- "Un réseau local LAN (i.e. Local Area Network)
- Le réseau étendu WAN (i.e. Wide Area Network)

Réseaux dans les jeux

Les sujets de préoccupations : (RV en M1 pour les détails)

- Topologie du réseau: client-server vs. peer-to-peer
- Programmation réseau : objets distribués ou envoie de messages (comment chaque joueur perçoit sa partie du jeu)
- Quel protocole d'échange utiliser ? tcp, udp, udp sécurisé
- Limitation de la bande passante
- Limitation de la latence
- Cohérence des données (dans le cas objets distribués)
- Détections des fraudes (tricherie)

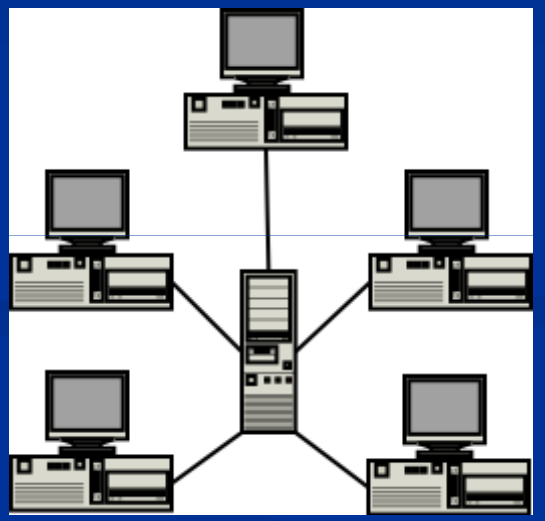
Peer-to-peer
(P2P ou « poste à poste »)



Ex : Utilisé dans Age of Empire

Client-Server (1)

- 2 parfums
  - Serveurs ad-hoc : n'importe quel joueurs peut jouer ce rôle (et changer en cas de déconnection)
  - Serveurs dédiés (ex : MMOG)
- 2 types de clients:
  - Les clients gèrent leur propre monde, le serveur rassemble et redistribue les états des joueurs (QuakeIII / Half-Life)
  - Les clients font des demandes d'infos au serveur, les simulations sont faites sur le serveur: utiles pour des clients « légers », e.g., tels mobiles et MMOG persistants (le monde continue d'exister entre deux connexions)



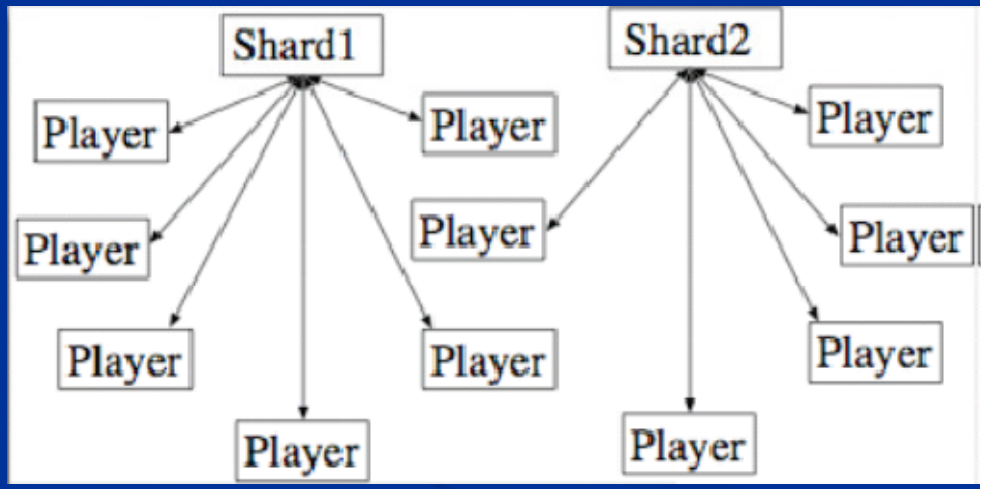
Client-server (2)

- Avantages :

- Inconvénients :

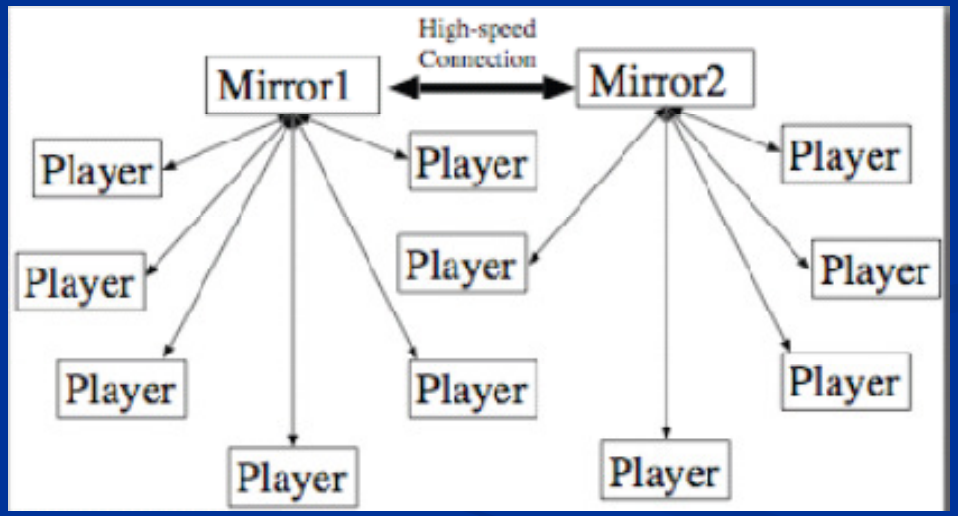
Architecture des serveurs de Jeux Online Massivement Multi-joueurs (MMOG) (1)

- L'environnement de jeu est répliqué sur chaque serveur (Shard)
  - Chaque serveur contient une réplique indépendante du monde (ex : serveur France, UK, USA…)
  - Les joueurs choisissent un serveur spécifique
- Ex : la plupart des MMORPG



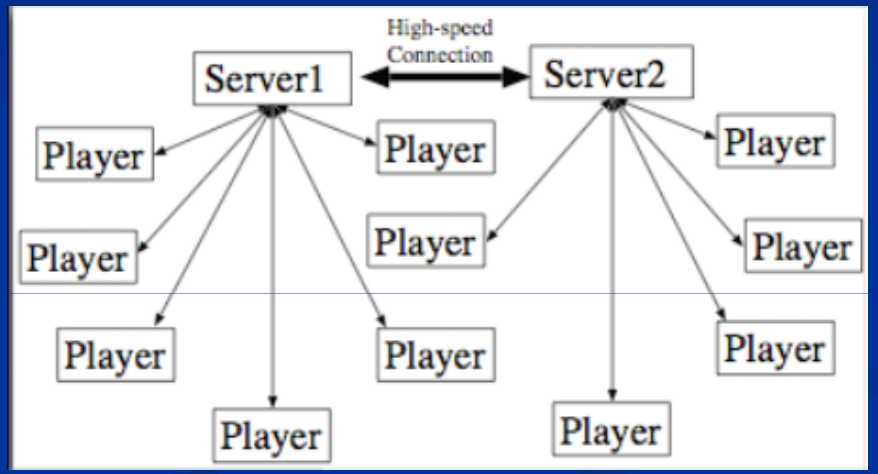
Architecture des serveurs de Jeux Online Massivement Multi-joueurs (2)

- L'environnement de jeu est aussi répliqué sur chaque serveur (Miroir)
  - Tous les environnements sont synchronisés
  - Les joueurs se voient tous à travers les miroirs
- Les miroirs doivent être maintenus cohérents entre eux



Architecture des serveurs de Jeux Online Massivement Multi-joueurs (3)

- L'environnement est séparé en régions
  - Chaque région est disponible sur un serveur différent
  - Ex: "Second Life" (sims online)
- Les serveurs doivent être maintenus cohérents entre eux



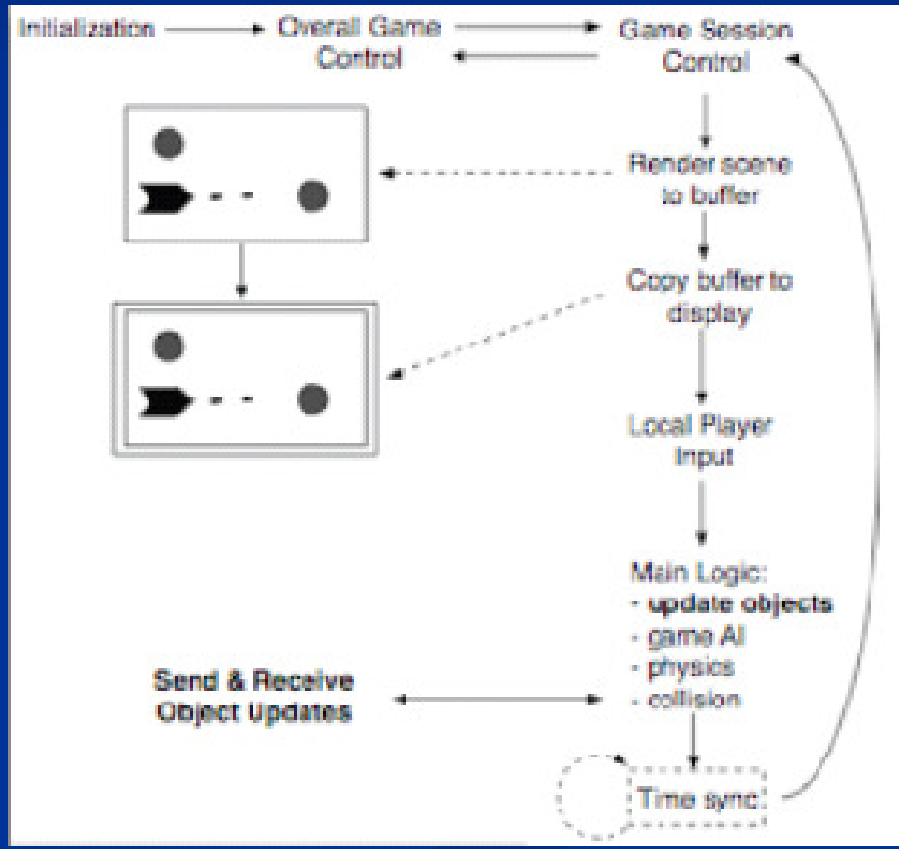
Modèle de calcul distribué

- Jusque là : architecture réseau
- Maintenant on parle de programmation (il y a des liens mais les deux problèmes sont indépendants)

- En général, votre entreprise aura son modèle de calcul préféré et fournira elle-même les bonnes librairies (catalogue de fonctions) pour implémenter le modèle
- Deux parfums :
  - Communication d'objets haut niveau (par exemple codés en C++)
  - Envoie de messages simples

Modèle de calcul distribué
Objets distribués

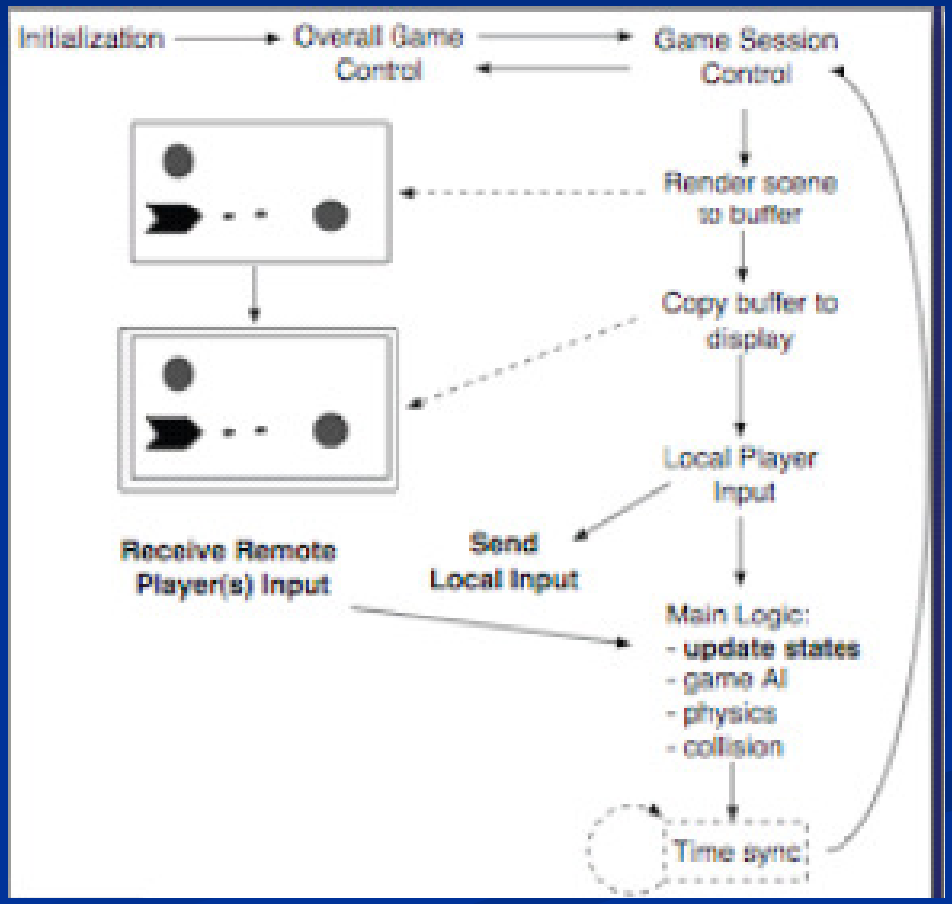
- Les personnages et l'environnements sont des "objets"
- Les interactions des joueurs (input) s'appliquent à des objets (stockés sur le serveur)
- Les changements sur les objets sont propagés aux autres joueurs à la fin de la boucle (après un « pas » de jeu)



Modèle de calcul distribué
échange de messages

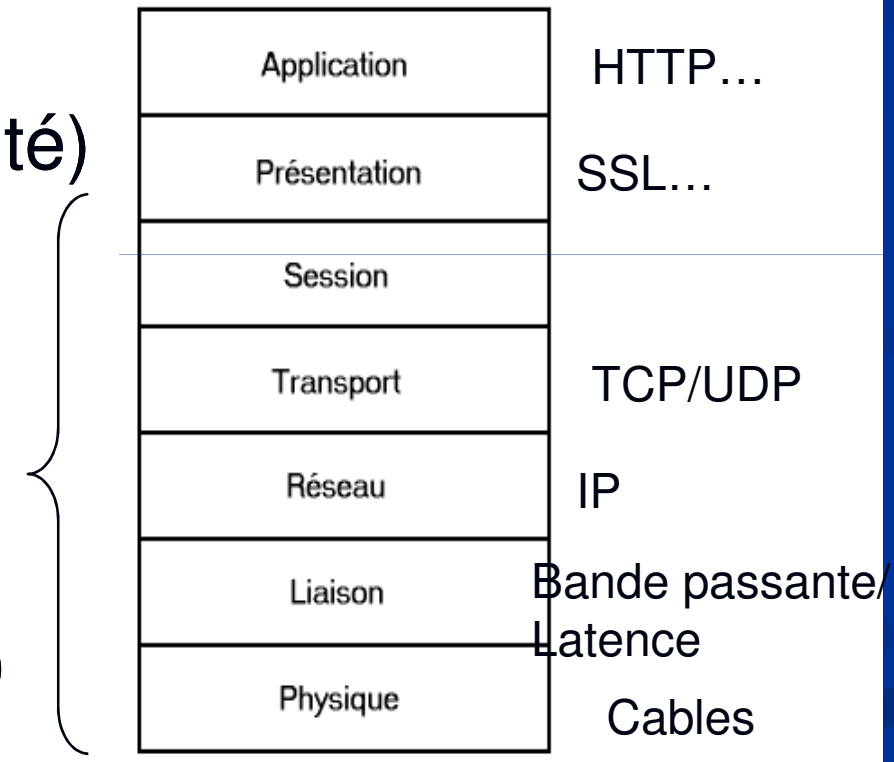
- Toutes les interactions des joueurs (appuyer sur un bouton ou tous les mouvements haut niveau) sont envoyés aux autres joueurs

- Chaque joueurs met à jour son propre état du jeu ou
- Le serveur met à jour l'état global du jeu et l'envoie à tous les joueurs



Quel protocole utiliser ?
Couches des protocoles OSI

- Possibilité d'agir (programmes, sécurité)
- Pas d'améliorations possibles (sauf LAN)



TCP vs. UDP

- IP (« Internet Protocol ») envoie des paquets de données d'une source à une destination (identifiées par une adresse IP unique au monde) : taille max des paquets IP 64 ko, il sont fragmentés pour passer…
- TCP et UDP sont des protocoles de la couche transport (implémentés au dessus du protocole IP (couche réseau))
- TCP (Transmission Control Protocol) offre :
  - Remise sure (intégrité des données transmises)
  - Retransmission et remise en ordre des paquets
  - Contrôle de la congestion (bande passante)
- UDP (User Datagram Protocol) offre :
  - Latence minimale, Bande passante supérieure
  - Possibilité perte de données
  - Pas de retransmission, les paquets doivent impérativement arriver dans l'ordre pour être compris
  - Pas de contrôle de la congestion : pas de qualité de service

Quel protocole utiliser pour les jeux ?

- Nécessités du jeu :
  - Les paquets qui arrivent en retard ne sont peut être plus utiles
  - L'information perdue peut quelque fois être interpolée (ex : pendant une course de voiture) mais des statistiques de perte de paquets sont nécessaires (savoir combien perd de paquet en moyenne)

- Utilisation d'UDP dans les jeux : créer un UDP sûr = prendre des avantages à la carte
  - On peut mettre des priorités sur les données
  - On peut s'assurer de l'intégrité des données
  - Peut gérer des données redondantes (plus précis) ce qui n'est pas possible en TCP à cause de la lenteur du transport
  - On peut envoyer des valeurs absolues, pas des intervalles (le temps de transmission des paquets devient négligeable donc on peut identifier la position des paquets à un instant donné)
  - Si on envoie des intervalle, on peut s'arranger pour se synchroniser souvent (pour savoir exactement où sont les paquets)
  - On peut contrôler la congestion en cas d'envoie de gros paquets de données

Limitation de la bande passante (1)

- Qu'est ce que la bande passante?
  - Quantité de données envoyées/reçues dans un medium pendant un temps donné (en bits/s)
- Quelle est l'information envoyée ?
  - Dépend du modèle de calcul choisi : des objets (distribués) ou des messages (envoyés)
  - Les états du jeux : coordonnées, statuts, action, mouvements articulaires, dégâts….
  - Les touches (interactions) appuyées, commandes, mouvement…
- Ex : pour AoE: 1 paquet toutes les 1.5-2 sec, jusqu'à 3-4 commandes/sec pendant les combats… Bande passante d'au moins 56 Kbps pour avoir un jeu fluide !

Limitation de la bande passante (2)

- L'utilisation de la bande passantes est en général TRES optimisée dans les jeux. Même avec un « chat » audio, on n'utilise rarement plus de 8 Kops
- La gestion de bande passante ne devrait pas un être problème sérieux (même si avec 8 Kops, on ne peut faire jouer que 4 joueurs sur un modem à 28.8 Kops)
- Pourtant, en donnant la possibilité aux joueurs de créer leurs propres objets et mondes, la bande passante devient à nouveau un problème d'où :
  - envoie continue des données sans synchronisation
  - pre-fetching (récupération des données avant le besoin de l'utilisateur)

Limitation de la latence

- Qu'est ce que la latence ?
  - le temps que va mettre la ligne pour être opérationnelle
  - ex : télé satellite, toujours 8s de latence mais vision continue (décalée de 8s) car BP très importante
- Si la taille des infos à envoyer est grande : pas de problème de latence (ex : télé satellite)
- Sinon, c'est important ex : msn avec une latence de 8s ☹
- Ex de latences :
  - RTS (« Real Time Strategy »): <= 250 ms pas décelable, 250-500 ms jouable, > 500 ms décelable
  - FPS (« First Person Shooter ») <= 150 ms préféré
  - Course de voiture: < 100 ms préféré, 100-200 ms lent, >= 500 ms, plus de contrôle de la voiture…
- Les attentes du joueurs peuvent s'adapter à la latence
- Il vaut mieux être lent et régulier que rapide par à coup…

Programmeur réseau jeux ?

Conclusion

- A retenir :

- Merci à :
  - **Jonathan Schaeffer** (“Learning to have fun”, Games Research Group, Département d’Informatique de l’Université d’Alberta, Canada, jonathan@cs.ualberta.ca)
  - **Pieter Spronck** (“http://www.spronck.net/”) : chercheur en Intelligence artificielle pour les jeux, Université de Tilburg, Pays Bas.
- Liens intéressants :
  - http://aigamedev.com/

Après ce cours…

- Machines Abstraites
  - machine à registres
  - machine de Peano
  - machine de Turing
- Arbres
  - Définitions
  - Données arborescente
  - Structure d'un espace de recherche
    - Recherche en profondeur d'abord
    - Algorithme MinMax
    - Recherche de stratégies