Video game

Graphics Programming

Eric Cannet

But de ce cours

- Le monde du jeu vidéo
- Des moteurs existant
- Les consoles next gen
- Des concepts

Le monde du jeu vidéo

Quelques chiffres

- PDF du SNJV datant du Mercredi 19 octobre 2011

Le jeu vidéo en France

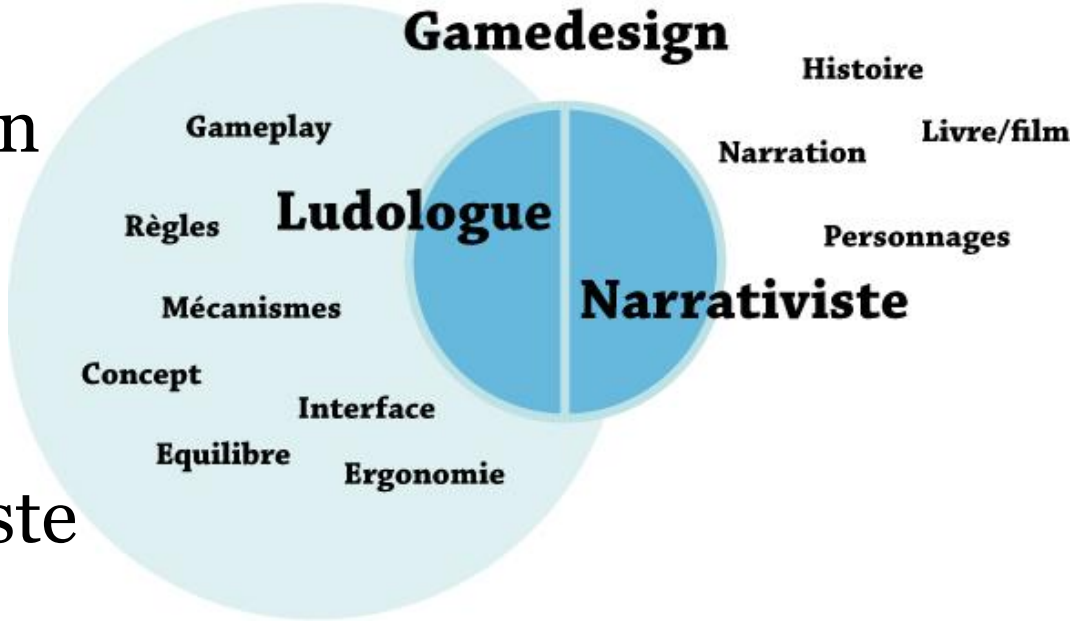
- C’est la crise ! Beaucoup de studios ferment
- Mais rassurez vous il en reste encore.
- Certains secteurs sont porteur et recrute.
- Le SNJV a publié une nouvelle étude (Octobre 2012)

Management

- Directeur de production
- Chef de projet
- Directeur marketing
- Chef de produit

Design

- Directeur de création
- Lead game designer
- Game designer
- Level designer
- Scénariste/dialoguiste
- Ergonome
- Designer sonore



Image

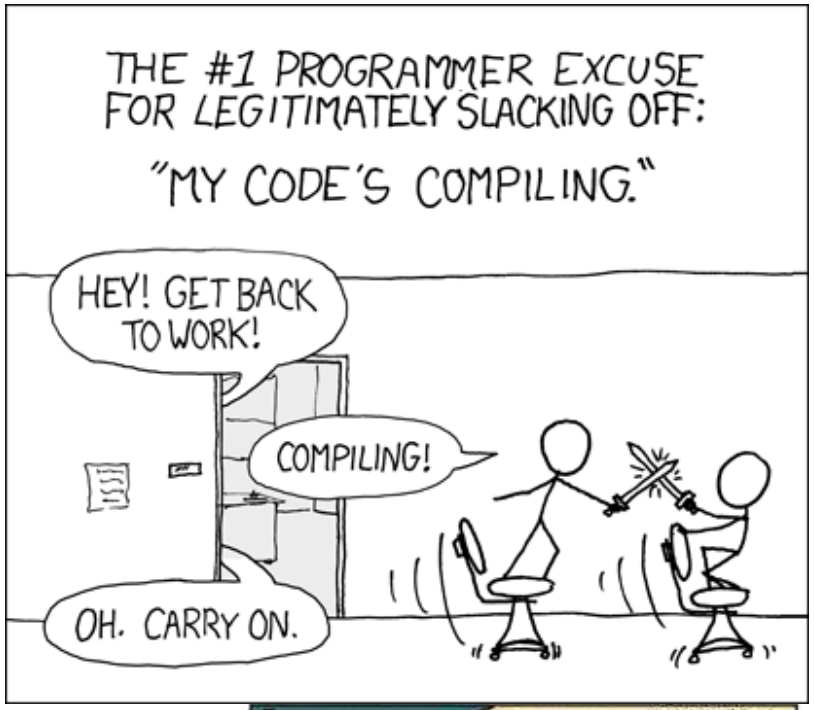
- Directeur artistique
- Concept artist
- Lead graphiste
- Graphiste 2D
- Modeleur/textureur 3D
- Animateur
- Spécialiste graphique

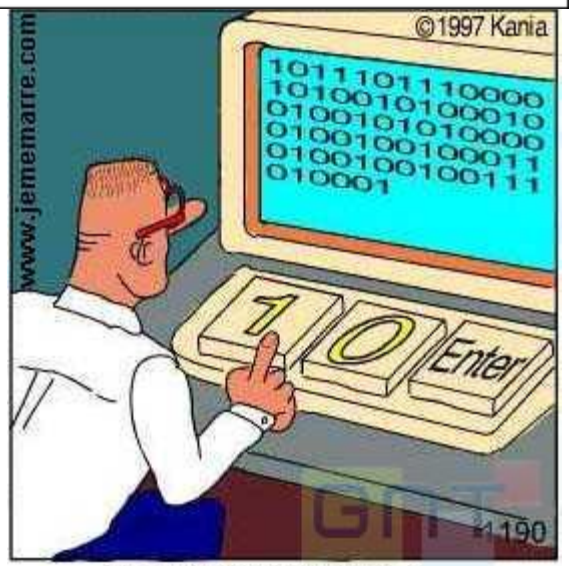
Transverse

- Testeur/QA
- Chargé de localisation
- Data manager/Asset manager
- Community manager
- Data analyst

Technologie

- Directeur technique
- Lead programmeur
- Programmeur moteur
- Programmeur gameplay
- Programmeur spécialisé





Le vrai programmeur ...

Game Engine

Game Engine ?

- Qu’es ce qu’il y a dans un moteur ?

- Qu’es ce qui important dans un moteur ?

Game Engine

- Un game engine regroupe (liste non exhaustive):
  - Abstraction du matériel
  - Moteur 3D, affichage
  - Inputs
  - Moteur Physique
  - Audio
  - Gestion de la mémoire
  - Accès au système (aux fichiers)
  - Bibliothèque de math (classique et vectoriel)
  - Gestion de thread, task
  - Réseaux
  - Une couche pour pouvoir faire du gameplay
  - Des outils

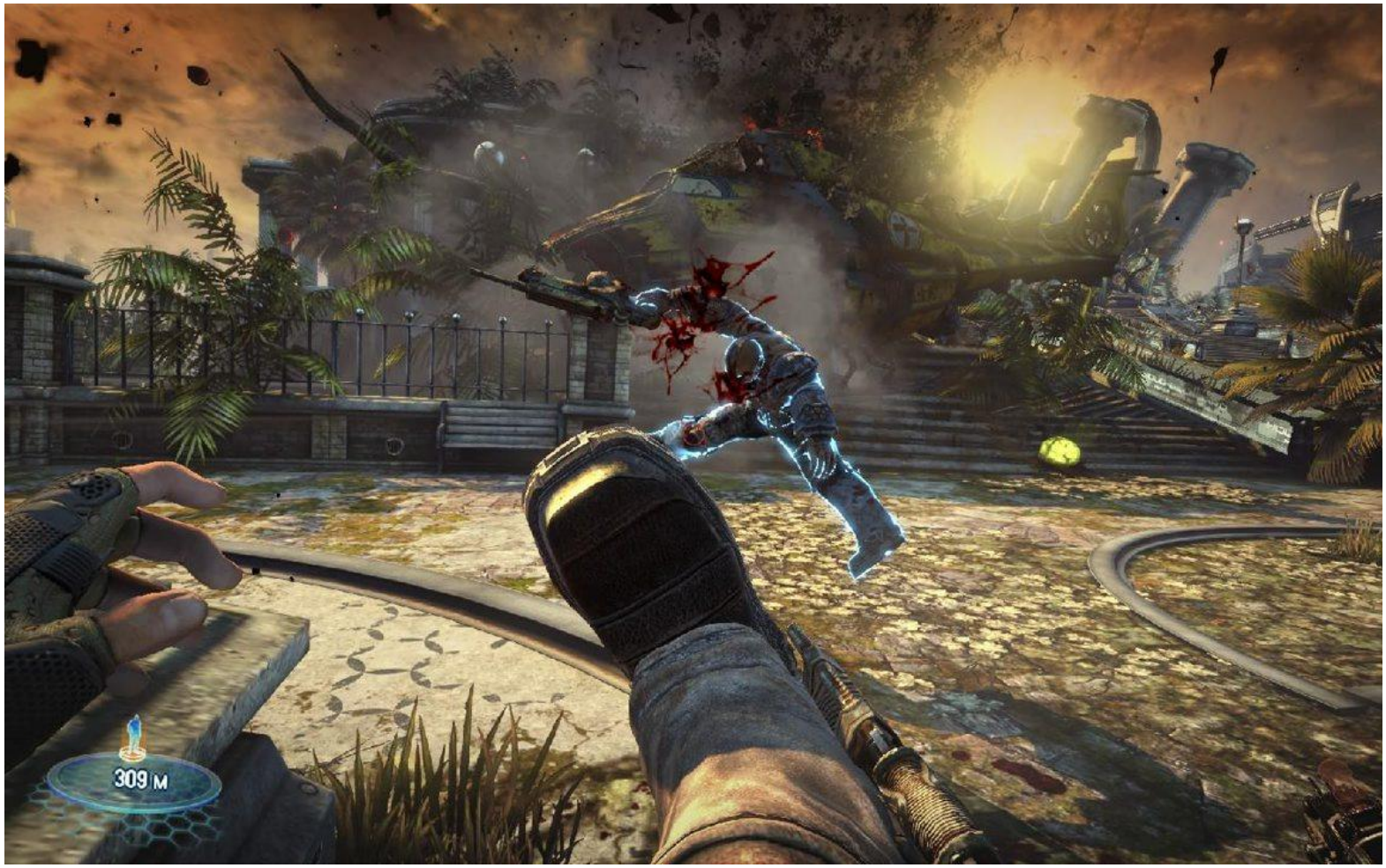
Unreal Engine

- Développé par Epic Game
- Supporte: PC/Mac OS, X360, PS3, PSVita, Tablette/Téléphone, Web browser, Wii U

- Utilisable gratuitement via l'UDK

- Incontournable tellement il est utilisé

Unreal Engine Image



Cry Engine

- Développé par Crytek
- Supporte :PC, X360 et PS3

- Notable par la qualité des rendus extérieur et sa gestion de la GI

- Utilisable gratuitement

Cry Engine Image

Id Tech

- Développé par Id Software
- Supporte: PC/Mac OS/ Linux, X360, PS3

- Remarquable car John Carmack et les mega texture, OpenGL

Id tech Image

Unity

- Développé par Unity Technologies
- Supporte: PC/Mac OS, X360, PS3, Wii, Wii U, Tablette/Téléphone, Web browser

- Remarquable car moteur montant, les plateformes qu'il a commencé à supporter

- Utilisable gratuitement

Unity Image

Frosbite Engine

- Développé par DICE
- Supporte: PC, X360, PS3

- Remarquable pour sa physique, l'absence de DirectX 9, et Whaou

Frosbite Engine Image

PlayAll

- Développé par des studio français dont il reste que Kylotonn Games
- Supporte: PC, X360, PS3, WiiU,
  en cours: PS4 et prochainement XBoxOne

- Remarquable car Français ;) et pour plein d'autres raisons.

PlayAll Image

PlayAll Image

PlayAll Image

CyaTech

- Développé par Cyanide
- Supporte: PC

- Orienté next gen, mais aucune partie game et jeune

Yet Another Game Engine

- Il en existe plein d'autres:
  - Source Engine (Valve)
  - Creation Engine (Bethesda)
  - Anvil (Ubisoft)
  - Real Virtuality (Bohemia Interactive Studio)
  - IW Engine (Infinity Ward)
  - Glacier Engine (Io Interactive)
  - Luminous Engine (Square Enix)
  - RedEngine (CD Projekt)
  - Rage (Rockstar Avanced Game Engine)

Creation Engine

Anvil

Real Virtuality

Consoles

Avantage/Inconvénient d'une console

- L'architecture matériel est toujours la même
- Moins de risque de piratage
- Les outils
- Le prix (l'avantage) et le prix (l'inconvénient)

PS3

Architecture matériel

- 1 PPE et 8 SPE(Synergistic Processing Elements):
  - 1 + 6 = 7 core au total !
  - 3.2 GHrz et Big endian
- Une carte Graphique nVidia RSX
  - Peu performante
  - 256 Mo
  - On peut mapper de la RAM.
- 256Mo de RAM (512 sur les Kit)
- 1 disque dur et un lecteur Blu-Ray

Kit PS3 et PS3 de test

- La PS3 de test permet uniquement de lancer le jeu

- Le kit PS3 ou PS3 de debug permettent de tout faire:
  - Debugger
  - Profiler
  - GPAD(er)

Compilation

- SNC C/C++ : compilateur de sony:
- GCC : heu vous connaissez ?

- Intégration à Visual Studio et permet la compilation partagé

- Le choix se fait via un define dans visual studio

Lancement

- Connecter vous à une PS3 via le target manager (kicker la personne connecté si il y en a une)
- Il suffit d'appui sur F5 dans Visual et tout se fera tout seul ☺
- L'éxécutable est de type (ORC heu) ELF
- Par contre le debugger est séparé et s'appel ProDG
- Pas besoin de copier des données sur la PS3

Outils

- Il existe une ribambelle d'outils:
  - Target Manager
  - SNDBS
  - ProDG
  - GPAD: permet de capturer une image et de voir tout ce qui se passe sur le GPU et de faire du profiling. Equivalent de PIX sur PC
  - PSgen
  - Tuner: Profiling PPU et SPU

OpenGL / GCM

- Petit rappel:
  - Pas de shader unifié
  - Pas de registre pour les pixel shader, le code est patcher
  - Carte graphique très peu performante

- Deux possibilités:
  - OpenGL
  - GCM

OpenGL

- Nommé PSGL
- PSGL correspond à :
  - OpenGL ES 1.0
  - Shader en CG
  - OpenGL ES 1.1 extensions (VBO, FBO, PBO, Cubemap)
  - texture extensions (FP, DXT, 3D, NPO2, Aniso, Depth, Vertex Textures)
  - primitive/rendering extensions (Instancing, Primitive Restart, Queries, Conditional Rendering)
  - synchronization extensions (Fences, Events)
  - SCE performance extensions (TextureReference, AttribSet)

GCM

- Plus bas niveau que OpenGL
  - cellGcmSetDrawArrays
  - cellGcmSetFragmentProgram

- Utilisé en général avec la bibliothèque EDGE qui utilise les SPU pour traiter les données

Edge

- Bibliothèque fournie gratuitement développé par Naugthy dog (et sony vue que c'est une filiale) (Uncharted)

- On créer un job qui est exécuté par les SPU
- EdgeAnim:
  - Mélange plusieurs animation par squellette
- EdgeGeom:
  - Fait du culling (en modifiant la géometrie)
  - Gère l'animation par squelette

TRC

- TRC : Technical Requirements Checklist
- Ensemble de points technique qui doivent être respectés:
  - Le jeu ne doit pas cracher ou entrer dans un unresponsible state
  - Le pshome doit s'afficher dans un certains temps quelque soit l'endroit où on se trouve
  - On ne doit pas crasher ou autre lors de l'éjection du Blue Ray

X360

Architecture matériel

- 3 core PPE(PowerPC Processing Element) hyperthreadé:
  - 6 core au total !
  - 3.2 GHrz et Big endian
- Une carte Graphique ATI Xenos
  - Shader Unifié
- 512Mo de RAM
  - partagé entre le CPU et le GPU
- 1 disque dur et un lecteur HD-DVD

La XBox de test / Kit XBox

- Ce que l'on peut faire avec une test :
  - Emulation DVD
  - Lancer

- Ce que l'on peut faire en plus avec un kit:
  - Debugage Kernel et via Visual Studio
  - Pixer (utilisé pix, chercher c'est dans le dictionnaire….)

Lancement

- On lance comme pour windows, F5
- Visual Studio ne va copier que l'exécutable (qui a l'extension xex…)
- Toutes les datas doivent donc être copiées par le développeur.

- Pour cela il y a XBECopy. C'est un utilitaire en ligne de commande avec plein d'options très utile.

Debug et outils

- On debug via visual studio comme n'importe quelle projet Windows

- Quelques outils supplémentaires:
  - XbWatson : pour voir l'output de debug de la console
  - Pix : Comme pour PC mais en pas pareil☺. Permet en plus de faire des profiling de performance GPU et CPU
  - XbGameDisc : permet la création de DVD
  - Xbox Neighborhood
  - Et plein d'autres!

XNA/DirectX

- On peut utilise
  - C# et XNA
  - C++ DirectX

- Pas de pipeline fixe

DirectX

- Il s'agit d'un DirectX 9.c optimisé pour Xbox
- Pas d'abstraction logiciel : hardware abstraction layer (HAL).

- Plus D'objet COM mais les méthodes sont toujours là (AddRef, Release)

- 10 Mo Embedded Dynamic RAM (EDRAM) pour les différentes Render Target (back buffer, depth buffer, Render target)

Depth

- Floating point Depth Buffer(D3DFMT_D24FS8)
  - 1 le plus près et 0 le plus loin (1 – z) !
  - Valeur de clear : 1.0f
  - Fonction de comparaison : Greater / Greater equal

- Hierarchical Z / Stencil:
  - Test de Z effectué avant le Pixel Shader
  - Fait sur des blocs de 8x8 (configurable)

Command buffer

- Un peu comme les Display List en OpenGL.
- Il s'agit de commande directement interprétable par la carte graphique, on économise ainsi les appels au drivers

- En deux étapes:
  - Création (création d'un device, set des render target, puis la commande)
  - Lecture

- La création peut être faite sur Xbox mais aussi sur PC

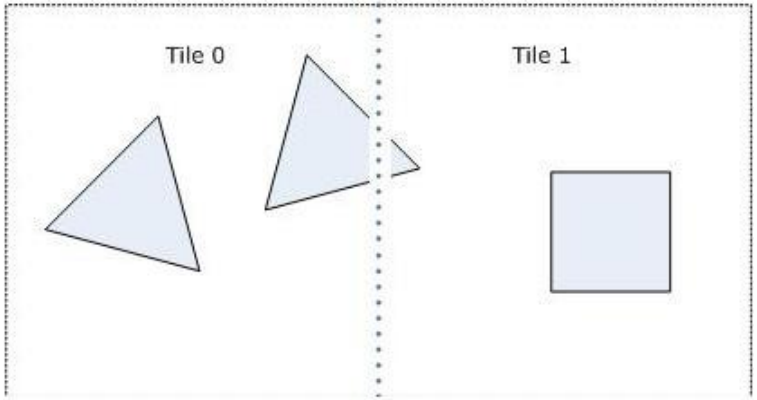
Tilling

- Sert quand les 10Mo de EDRAM n'est pas suffisant

- Divise l'écran en Tiles (Vertical ou horizontal)
- Quelques modification :
  - IDirect3DDevice9::BeginTiling
  - IDirect3DDevice9::EndTiling

Tilling image

- Enregistre automatiquement toutes les appels DirectX dans un command buffer
- Les rejoue une fois pour chaque tile



TCR

- TCR: Technical Certification Requirements

- Ensemble de point technique qui doivent être respecter:
  - Pas de plantage
  - Gestion de la connexion et déconnexion de profile
  - Gestion des storages
  - Pas d'image fixe pendant plus de 5 sec
  - Les sauvegardes doivent afficher un message
  - Pas de temps de chargement de plus de 60 sec
  - Arriver à la premier image interactive en moins de 20 sec
  - Ne pas avoir de déconnexion réseau temps que des conditions sont remplies
  - Ne pas overclocké le processeur et la RAM
  - Etc etc etc

Des pistes d'optimisation

- On ne peut faire un gros jeux sans un max d'optimisation:
  - Utiliser au maximum l'architecture de la console
    - SPU
    - Dual pipe
    - Massivement parallèle
  - SIMD: Single Instruction on Multiple Data
  - Les allocations et le dynamic cast sont très couteux

D’autres pistes

- CPU ou GPU Bound
- Répartir quelque chose sur plusieurs images
- Maintenant ce qui coûte le plus cher c’est n’est pas les instructions mais l’accès à la mémoire
  - Data Oriented Programming
  - Optimisation de Cache (Cache Miss, mais aussi Instruction Miss)

Des concepts

Multi-pass

- Comment coder pour différentes plateformes (Windows, linux, X360, PS3, WII, DS, PSP etc) ?
  - Define ?
  - Typedef ?
  - Héritage et méthodes virtuelles ?

Define

- On ne peut y échapper
- Bonne solutions pour des petits changement de code:
  - little ou big endian
  - Pour choisir un include
  - Define spécifique par platforme

Héritage

- Solution d'une architecture orienté objet « pure »
- Solution classique et naturel

- Quelques inconvénients:
  - Le coût des méthodes virtuelles
  - Si on modifie une méthode de la classe mère il faut modifié la classe fille. On voit le problème qu'a l'éxécution

Typedef

- Solution plus C:
  - On a une classe par plateforme et on fait un typedef de la bonne classe suivant la plateforme.
- On a plus les inconvénients de la méthode d'héritage et de méthode virtuelles.
- Inconvénient:
  - L'interface est définie par la compilation
  - Pas de comportement par défaut ou en commun

The winner is

- Alors la solutions ?

- Un mixe des trois:
  - Le define, on ne peut y échapper et sert pour la méthode du typedef
  - L'héritage pour avoir des comportements en commun et stocker des objets de type différent dans un même conteneur. Mais garder un minimum de méthodes virtuelle et éviter les dynamiques cast !
  - Le typedef au maximum et surtout pour du bas niveau

Render Loop

- Une boucle de rendu est la suite d'étapes de rendu pour avoir l'image d'afficher à l'écran.

- Cela peut être :
  1. Rendu des shadow map
  2. Rendu de la géométrie et de l'éclairage
  3. Rendu des postprocess
  4. Rendu du GUI

Forward Rendering

- Il existe plusieurs algorithmes classique de rendu.

- Nous avons déjà vue le Forward rendering
  - Cela consistait à calculer l'éclairage en même temps que l'on affiche la géométrie
  - Mais cela devient vite ingérable dès qu'il y a un certains nombre de lumières et de types d'objet

Deferred Rendering

- On affiche dans des RenderTarget toutes les informations pour faire l'éclairage en post process:
  - Couleur Diffuse
  - Coefficient Specular
  - Nomal(Bumpé)
  - Depth
  - Autres

- Cette ensemble de render target est appelé G-Buffer

Deferred Shading

- On calcule l'éclairage en post process. Par exemple, on affiche une sphère pour une omni.
  - A partir du z(depth) stocké dans le G-Buffer et des coordonnées du pixel en cours on est capable de retrouver la position du point en 3D
  - On peut ensuite calculer l'éclairage comme avant

- Complexité : Nb objet + Nb lumière

G-buffer: Speculaire

G-buffer: Depth

G-buffer: Normal

G-buffer: Diffuse

Image final

Alors deferred ou pas ?

- Avantages:
  - Complexité plus faible et donc possibilité d'avoir beaucoup plus de lumières
  - On a déjà des informations pour les post process
  - Clean
  - On ne calcul plus d'éclairage pour des pixels qui seront écraser par la suite
- Inconvénients :
  - Lourd en mémoire (beaucoup de RenderTarget)
  - Lourd en bande passante
  - Problème des objets blendés (semi transparent)

Ligth Prepass ?

- Reprend l'idée du Deferred Shading.
- Nommé aussi Deferred ligthing

- En 3 passes:
  - Rendu de la géométrie dans le Gbuffer (Normal, coefficient de specular et depth uniquement !)
  - Rendu des informations d'éclairage
  - Re-rendu de la géométrie en utilisant l'éclairage calculé précédemment

Litgh Prepass ou bien ?

- On a un gros gain en render target puisque le G-buffer passe à seulement 3 informations compactable

- On a une grosse perte en terme de traitement de géométries puisqu'on la rend 2 fois

Trop des barres

- Voici un bar profiler d'un boucle de rendu en ligth pre pass:

- Sur PS3 la géométrie est traitée par les SPU
- Il existe aussi un double speed quand on ne rend que la profondeur
- Une optimisation possible?

Game Loop

- La boucle de jeu correspond à ce qu'il ce passe pendant une image.

- Il y a pas que l'affichage:
  - Traitement des inputs, le gameplay
  - Calcul de la physique
  - Calcul de l'IA
  - Affichage

- Tout ceci peut être fait avec du parallélisme

Du game en barre

- Les moteurs gèrent un système d'étape:
  - Manage
  - PostManage
  - Update
  - Post Update
- Ces étapes sont exécuté dans un ordre bien précis sur tous les objets

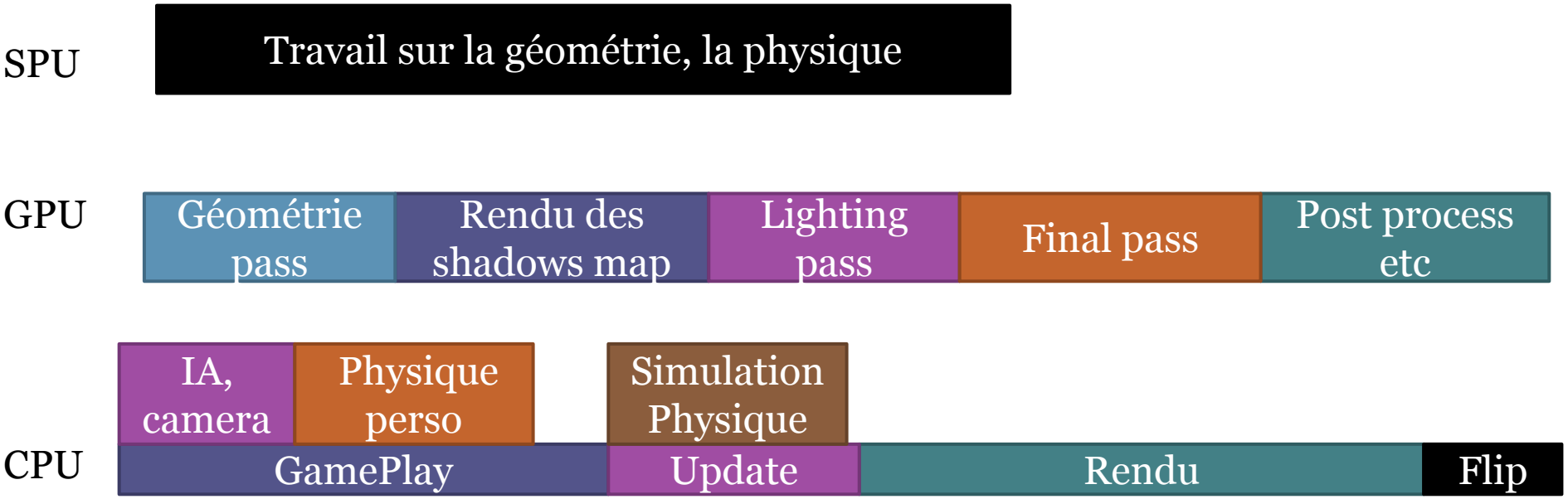
Last night a DJ saved my life

- Somme nous à une image près pour:
  - L'affichage ?
  - L'IA ?
  - La physique ?

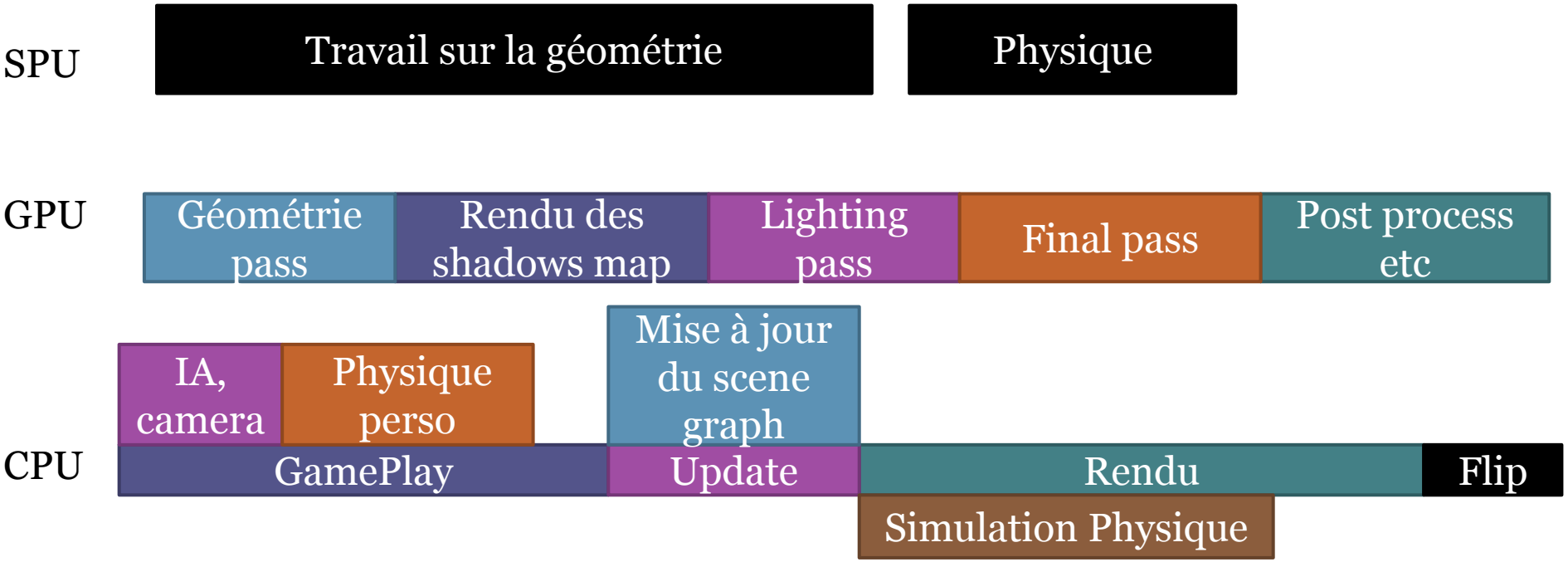
- Il y a souvent une image de retard. C'est-à-dire que l'on dessine l'image N-1 lorsqu'on calcul l'IA, la physique, le gameplay de l'image N

Optim quand tu nous tiens

- Une autre optimisation:



Et foila


SPU
Travail sur la géométrie
Physique
GPU
Géométrie pass
Rendu des shadows map
Lighting pass
Final pass
Post process etc
CPU
IA, camera
Physique perso
Mise à jour du scene graph
GamePlay
Update
Rendu
Flip
Simulation Physique

Graphe de scène

- On a un ensemble d'objet à afficher. La structure de données qui la contient est appelé scène graphe

- Il y en une hiérarchie. Un nœud du scène graphe à un parent et hérite de la matrice du parent
  - Matrice final du nœud = matrice du nœud * matrice du parent
  - Permet d'attacher un objet à un autre

Exemple de graph

- Certaines structures de données permettent d'accéléré des traitements:
  - BSP tree
  - Quadtree, Octree
  - KD-tree

- Enfin dans certain cas:
  - Dice à travail dessus. Premier test qu'ils ont fait sur leur nouveau système à été un bête tableau. Le résultat à été: plus rapide que l'ancien système. Pourquoi ?

Actor

- D’un point de vue gameplay un niveau est composer d’acteur
- Un acteur n’a aucun comportement.
- Ces acteurs vont avoir des composants:
  - Mesh
  - Physique Statique
  - Herbe
  - Trigger
  - Fog

Component

- Ces composants on une méthode par étape (Update,Manage) d'une frame et peut en avoir d'autre:
  - PostObjectInit
  - EnterCutScene
  - LevelLoaded

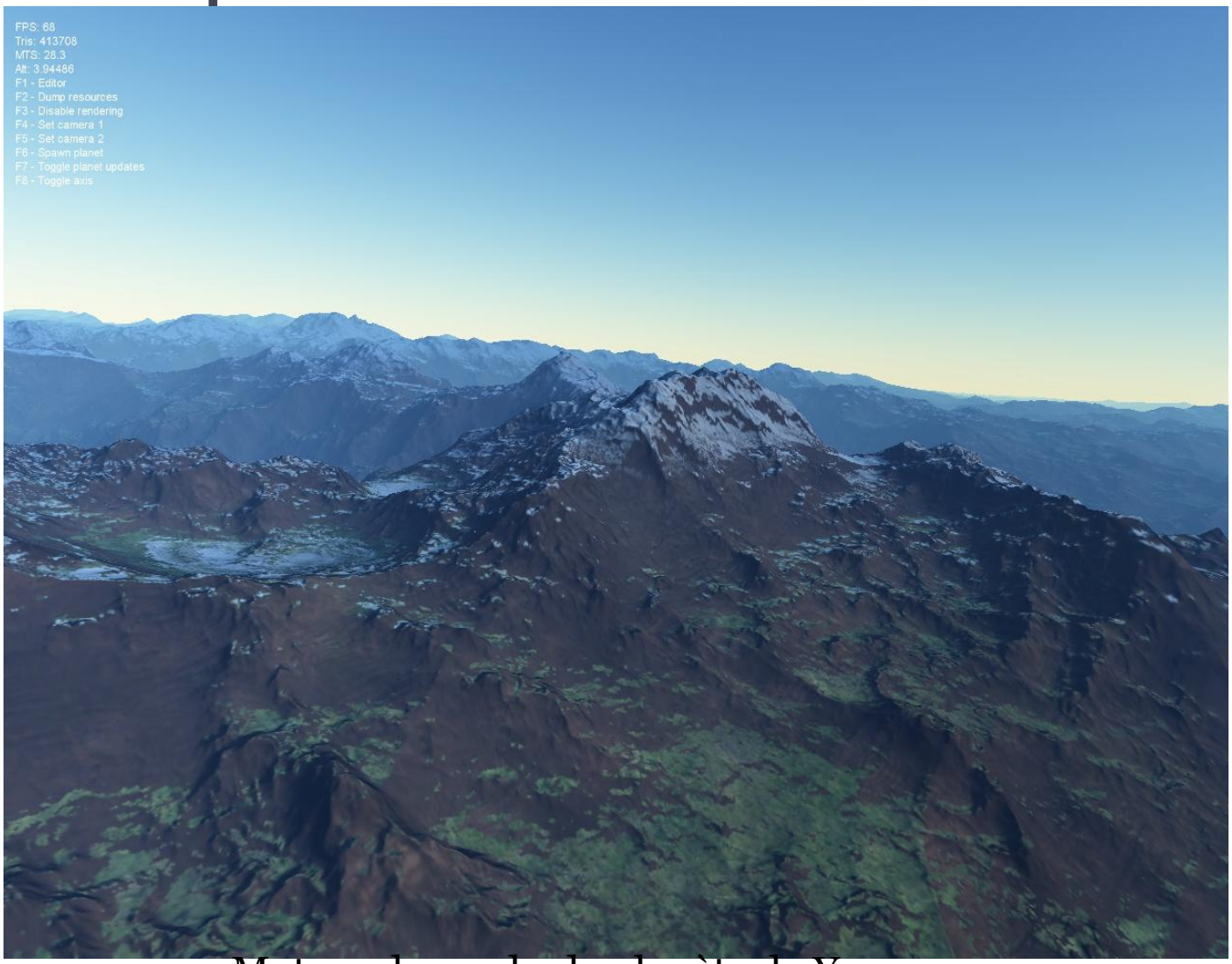
- Il gère aussi la configuration des objets moteurs qui vont derrière

Conclusion

- Le prochain cours : C’est un TP

- Des questions ?

Technique



Moteur de rendu de planète de Ysaneya

WiiU


Wii U
X
Y
A
B
START
SELECT
HOME
MIC
BATTERY
CONTROL
POWER
EJECT
SYNC

Architecture matériel

- 3 core PowerPC “Expresso” hyperthreadé:
  - 6 core au total
  - 1,24 GHrz et Big endian
- Une carte Graphique « GPU7 »
  - Shader Unifié avec geometry shader et une sorte de tesselation
- 1GoMo de RAM
  - partagé entre le CPU et le GPU + 34Mo de Edram (MEM1)

Gamepad

- C’est juste une manette avec un écran tactile.
  - Ce qui veut dire qu’aucun calcul ne peut être fait avec
  - Les images sont fait du côté de la WiiU et on envoi le buffer au gamepad
- N’est pas multipoint!

Kit WiiU, WiiU Reader/Writer

- Le Kit WiiU permet de tout faire
- La WiiU writer permet uniquement de graver des blue ray WiiU. Et le Reader uniquement de les lires. Impossible « de lancer en F5 » via une WiiU Reader
- Le GamePad peut être fillaire.

Outils

- Il existe des d'outils:
  - Cafe.bat
  - Spark
  - Green Hills multi
  - CPU profiler
  - Master editor
  - AppConfigTools

  - Des mouchoirs pour pleurer

En Vrac

- Bibliothèque graphique GX2
- Langage de shader différent: SLConverter

- Les guidelines : évasif et peu précis
  - Ejection du Disc
  - Passage au menu de la console