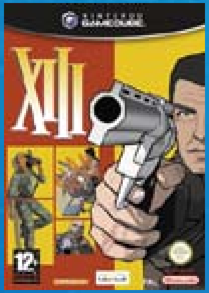
<< RETURN

The Impact of Middleware Technologies on your Game Development

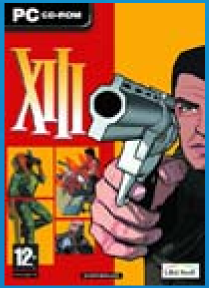
Julien Merceron
Worldwide Technical Director
Ubisoft Ent.

XBOX
SPLINTER CELL

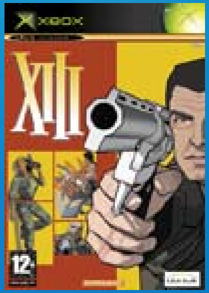
PC CD-ROM
Prince of Persia
Les Sables du Temps


Nintendo GameCube
XIII

PlayStation 2
Prince of Persia
Les Sables du Temps


PC CD-ROM
XIII

PlayStation 2
Beyond Good & Evil


XBOX
XIII

PC CD-ROM
Beyond Good & Evil

XBOX
Rainbow Six 3

PlayStation 2
SPLINTER CELL

SPLINTER CELL
PANDORA TOMORROW™

UBISOFT™

Introduction

- Topics:
  - Technology Design Process;
  - Project Life Cycle Design;
- Structure of the Talk:
  - From B.G.E to P.O.P;
  - From Unreal2 829 to Splinter Cell;
  - From Splinter Cell to Splinter Cell:PT;
  - Strategy highlight;
  - Practical Methodology, patterns analysis;
  - Q&A.

From B.G.E to P.O.P

(B.G.E = Beyond Good and Evil)

(P.O.P = Prince of Persia: The Sands of Time)

From B.G.E to P.O.P

- Level Architecture Differences:
  - POP is very linear, each location can only be visited at one time of day
  - World is completely dynamically loaded, less data in memory at the same time
  - The abilities of the Prince enable him to cover great distances in a short time, rooms must be bigger
  - Graphic style makes it harder to use the "all textures fit in GS" scheme used by BGE

From B.G.E to P.O.P

- How does this change the graphic engine?
  - No need to be able to adjust the ambient color dynamically as in BGE
  - No need to minimize the size of the graphic data by storing the meshes in indexed form. This form is suboptimal on PS2
  - Static meshes must render very fast to be able to display the bigger rooms we'll have in the game
  - Dynamic texture loading is needed (also required because of Dynamic Loading)

From B.G.E to P.O.P

- Choices we made - Move lighting and skinning on VU1
  - We moved skinning, ligthing, and all UV computing (chrome, projection for shadows) on the VU1
  - This gave us more CPU time for handling AI / Collisions / Etc…
  - We also had very high elements counts in some scenes so this gave us a little more breathing room.
  - PA showed that we were CPU bound not VU1/GS bound

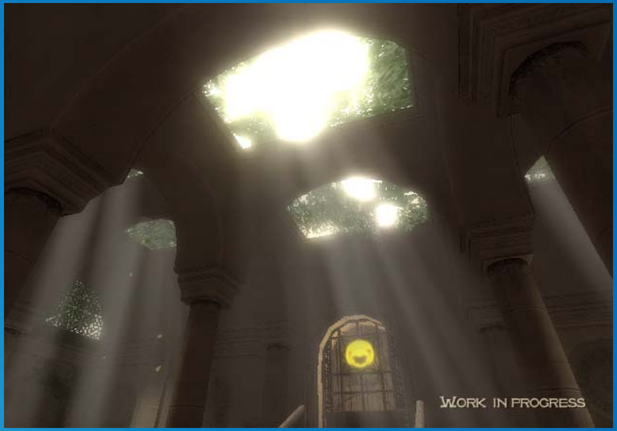
From B.G.E to P.O.P

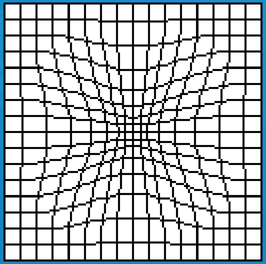
- Choices we made - Dynamic Texture Loading
  - Ended up being the one that involved the most changes to the engine
  - JADE doesn’t sort elements by texture, PS2 VRAM is not big enough so it must be done
  - Created a manager that kept various lists:
    - » Opaque, Cracks/Dirt, Transparent, Lightmaps, GFX, Glow Geometry

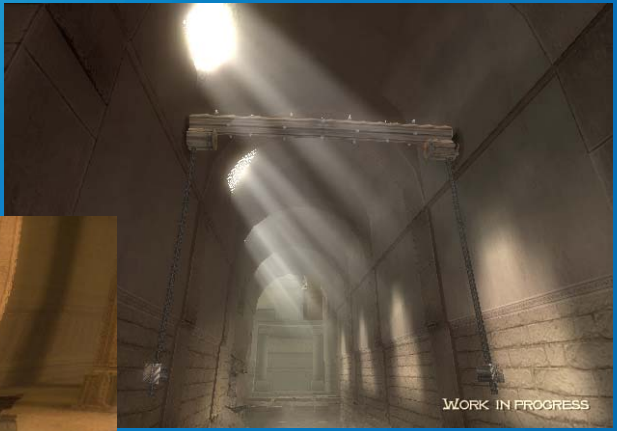
UBISOFT

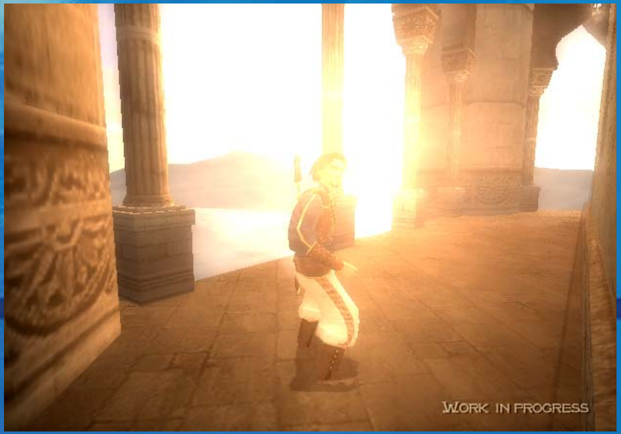
From B.G.E to P.O.P

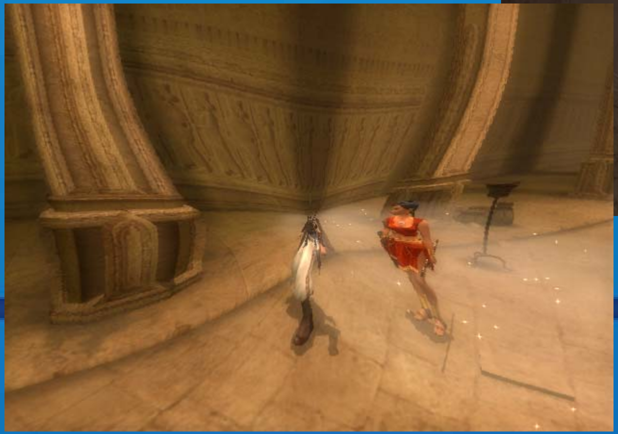
- Post Effects we added
  - Glow
  - GS Occluded Flares
  - Pinch







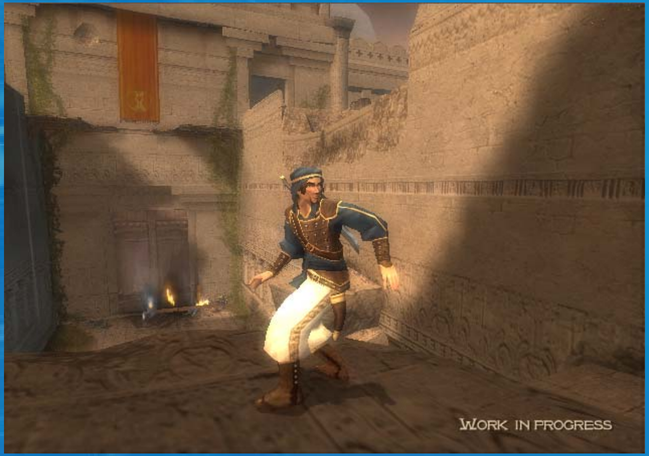


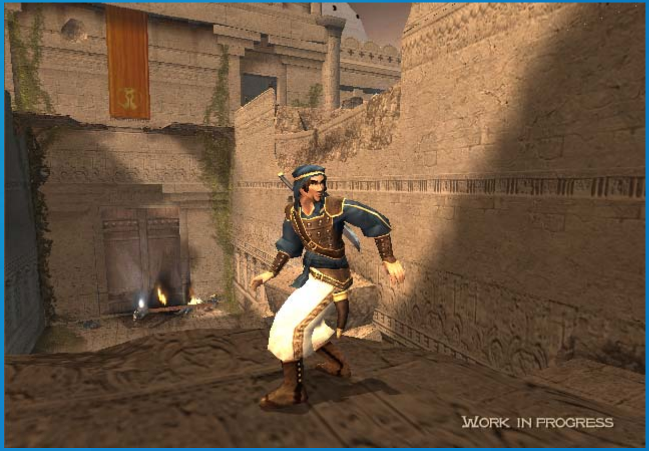


UBISOFT

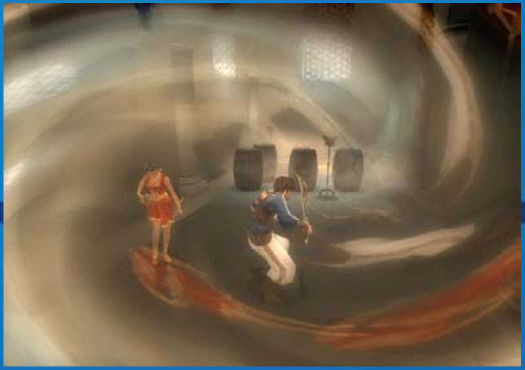
From B.G.E to P.O.P

- Other Post Effects
  - Smart Blur



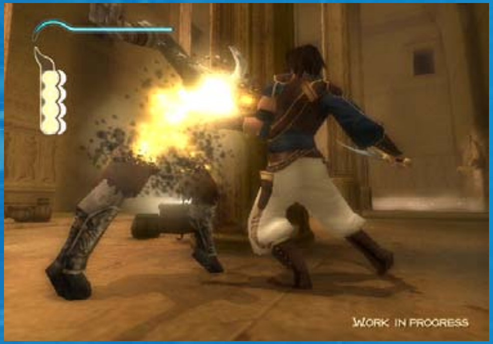


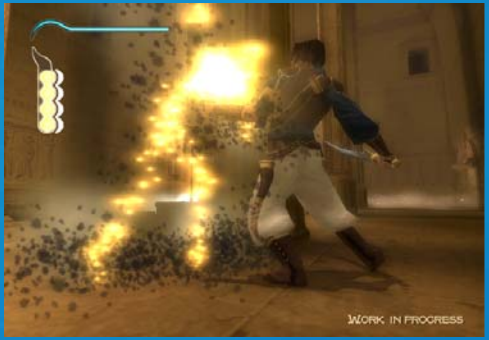
  - Spiral Blur



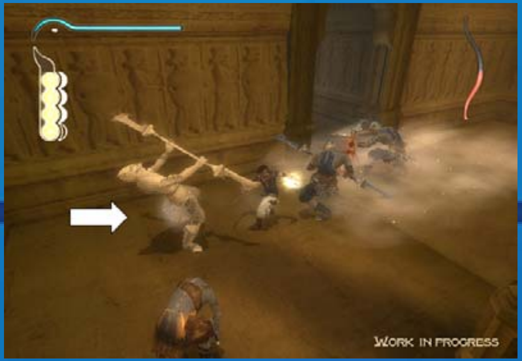
From B.G.E to P.O.P

- 3D Effects:
  - SandMan: Disintegration;



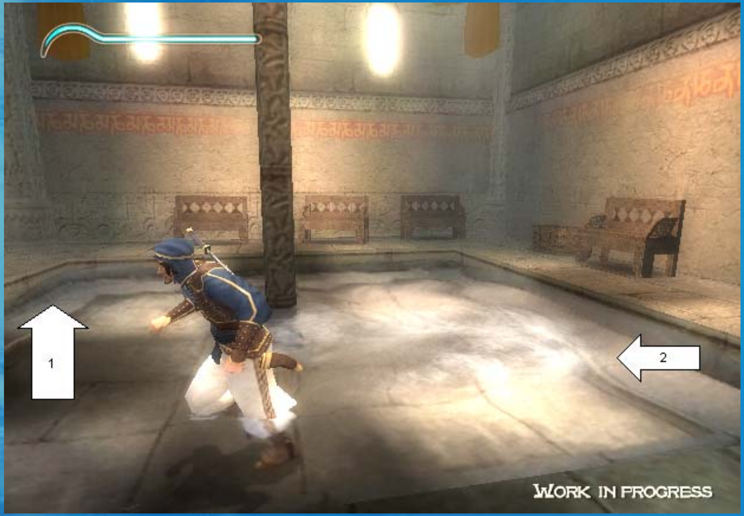


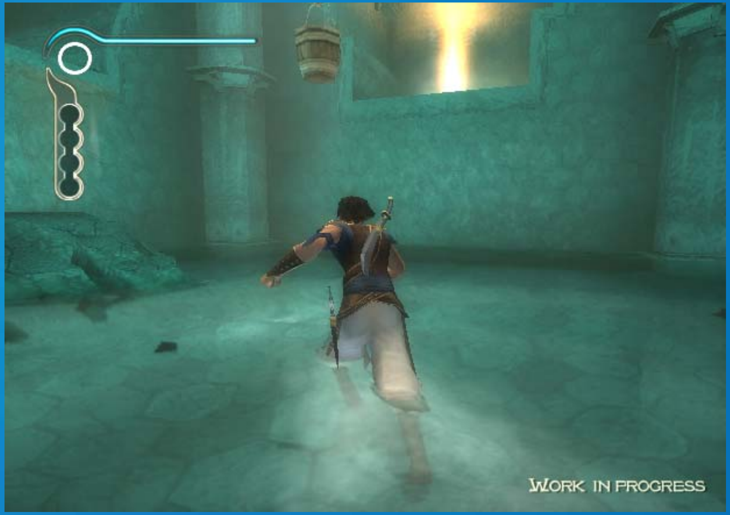
  - SandMan: Freeze



From B.G.E to P.O.P

- The Water Effect:





From B.G.E to P.O.P

- Workflow:
  - Faster binarization process (necessary for dyn load)
  - Atwinmon & Devlink support
  - MAX Exporters for animations (trl & gas)
  - Batch TRL & AVI Exporter in Max
  - Buildmonkey & binarization reports
  - Animation viewer
  - Automated build script (1 click = 1 iso)

From B.G.E to P.O.P

- Data Control
  - VSS mirroring
  - History (through VSS)
  - Show Differences (with group xml export)
  - Araxis merge integration for text files
  - Auto checkout (not really used)
  - Pending check-ins window
  - Token system
  - todo: DataMonkey

From B.G.E to P.O.P

- Interaction:
    - Dynamic bridges
    - Simple cloth simulation for characters (springs & rotation constraints)
    - Cloth simulation for drapes, flags, spider webs, etc
    - Rope simulation
    - Dynamic plants, trees
    - Wind system
    - Collision events: interaction with environment

From B.G.E to P.O.P

- Animation:
  - Synchronous animation blending (animix)
  - Partial asynchronous animation blending (animask)
  - Animation in 3DSMax
  - Partial action kits by action group (at binarization time)
  - Animation size optimization
    - » removed unused translation tracks (from Character Studio)
    - » store quaternion floats on 16 bits (unpacking with MMI & VU0)

From B.G.E to P.O.P

- Sound:
  - Integrated DARE
  - Asynchronous bankset loading
  - Microphone system
  - Inaudible sectors
  - Dialog manager (manage streams)

From B.G.E to P.O.P

- AI:
  - Added PostMessage functionality for AI (useful for rewind, message overflow, etc)
  - Added support for AI Functions
  - Removed references to script AI functions

From B.G.E to P.O.P

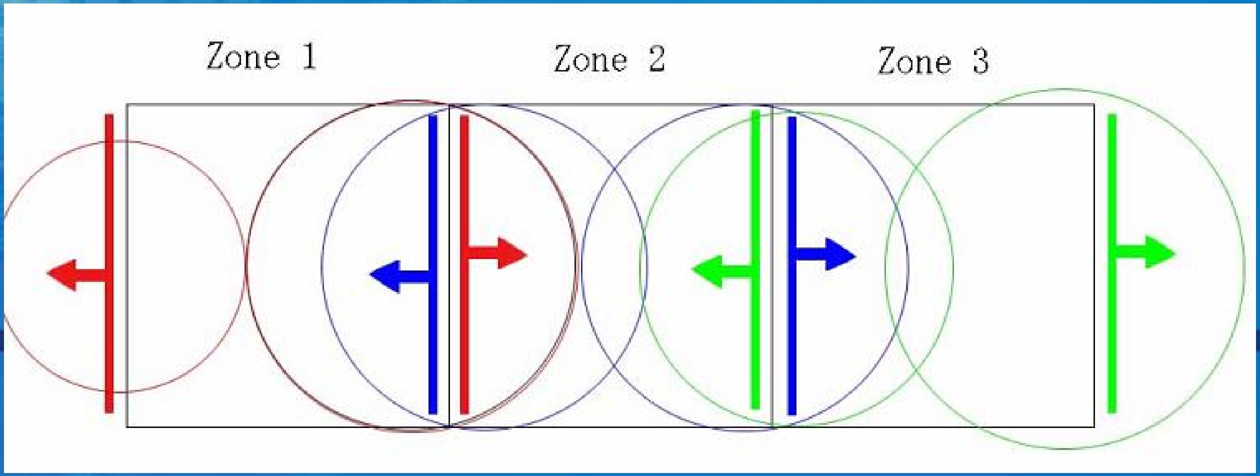
- Rewind/VCR feature:
  - A giant real-time built animation
    - » could be used for replays or demos.
  - Uses circular buffer for each track
  - AI callback
  - New kinds of keyframes:
    - » AI, Animation, Animix, Animask, Bridge, Dynamic/Col, Hierarchy, Rope, Vanim

From B.G.E to P.O.P

- The Dynamic Loading feature:
  - Seduced by Jak & Daxter…
  - A continuous world to increase immersion
  - "Just one more turn…"
  - More, More, More…
    - » More textures, objects, animations, sounds
  - Local ambient color with moving objects;
  - Fog settings;

From B.G.E to P.O.P

- The impact of Dynamic loading:
  - Edition is Zone-based;
  - A portal system manages the prefetching and the transition between zones;
  - Impacted on Modelisation, Artificial intelligence, Memory, Sound.



From B.G.E to P.O.P: PLC

- CONCEPT;
- PROTOTYPING
  - Starting prototyping ideas with CPA environment (Rayman3, Batman2, etc.);
  - Refined the needs for the game;
  - Unreal2 and B.G.E specs study;
    - B.G.E was closer to our needs.
  - Havok & the Splinter Cell game physics study;
    - SC physics – developed by Montreal - were closer to our needs.

From B.G.E to P.O.P: PLC

- PRE-PRODUCTION UNTIL FP:
  - 5 persons went to Montpellier to get JADE:
    - » Get & validate the Closing Kit;
    - » Learn how to use the technology;
    - » Create the documentation (yes!);
    - » Came back to Montreal and trained all the pre-production team.
  - Performance tests
  - Planning necessary re-architecture
  - POP technologies integration studies
  - Create the FP

From B.G.E to P.O.P: PLC

- PRE-PRODUCTION UNTIL FPP:
  - Integrate all the technologies;
  - Finalize first-drop pipeline;
  - Explain all the constraints to the team;
  - Create the FPP.
- PRODUCTION:
  - Have all your constraints clearly understood by the team;
  - Be primarily optimising the production pipeline to prepare the tools for the tuning phase;
  - Be optimising the engine :
  - Reduce as much as possible other developments.

From B.G.E to P.O.P: TD

- Tech Design:
  - Prototyping on CPA engine;
    - » Getting a better idea of what we wanted to do and the performances & features we needed;
    - » FP 1 was approved & next steps definition;
  - Going to FP2: polish & choosing the game engine:
    - » Polishing FP1 to bring FP2;
    - » Study of 2 solutions for the game: JADE & Unreal2;
    - » Hold-up in Montpellier ☺
    - » From a feature standpoint: JADE
    - » FP 2 was approved & next steps definition;
  - Time to start adapting JADE to POP & create FPP:
    - » Hardware projection with JADE levels;
    - » Planning & starting necessary re-architecture;
    - » FPP delivery;

From B.G.E to P.O.P

- Project Life Cycle – Postmortem:
  - We went fast through Prototyping & FP stages;
  - Most of the key feature development was finished for the FPP;
  - During production, the team focused on:
    - » Frame rate;
    - » Special effects;
    - » Production pipeline;
    - » Debugging stage & versioning;
    - » Giving more flexibility to the level designers.
  - P.O.P started 15 month after B.G.E and shipped at the same time.
    - » The pipelines of those games are different.

From B.G.E to P.O.P

- Tech Design – postmortem 1:
  - We had a Game Engine to start prototyping on;
    - » Created a reference HP;
  - We had access to an unlimited amount of technologies to choose from to build the prototypes & FPs:
    - » More flexibility to reach the targeted Game HP;
  - Technology & Game Engine choices were left to the development team:
    - » Maximize involvement & responsibility;
    - » Give them ownership over what they are doing.

UBISOFT

From B.G.E to P.O.P

- Tech Design – postmortem 2:
  - Middleware used were proven techniques, we knew how to integrate and use them.
  - We choose a robust Game Engine. It allowed us to build on that and therefore take more time and risks on our key features (rewind, dynamic loading, cutting edge special effects);

UBISOFT

From 829 to Splinter Cell

From 829 to Splinter Cell

- Acquiring Red Storm brought us into the FPS developer community;
- 2 important games were initiated at the same time in Montreal:
  - Tom Clancy`s Rainbow Six 3;
  - Tom Clancy’s Splinter Cell.
- We studied Game Engines:
  - From ID, Valve, LithTech and Epic;
  - Unreal2 was closer to our needs and internal practices;

From 829 to Splinter Cell

- We started with version 7xx
  - First levels were built in Unreal2 style within a few days;
  - Third person behavioral approach tested very fast;
    - » Animation & game play mechanics;
  - Dx8(.1) Graphic support integrated rapidly (including shader support, shadow maps & glow);
    - » First tests on the stealth system;
- We knew what the game would be…
  - … But there was so much to produce and develop that we continued even in production testing how far we could push the technology and the game play mechanics.

From 829 to Splinter Cell

UBISOFT

From 829 to Splinter Cell

- Heavy development in Production:
  - Was the big challenge;
    - » Light & shadow customization;
    - » Animation system enhancements;
    - » AI development.
- Synchronization with Epic:
  - until build 829;
    - » Then only bug correction integration.
- Reasonable choices
  - We integrated DARE (internal Sound Middleware);
  - We didn't integrate MORE (internal Animation Middleware).

UBISOFT

From 829 to Splinter Cell

- XBOX version in early Beta:
  - focused heavily on optimizations:
    - » Getting help from the ATG team helped!
  - started PS2 adaptation:
    - (see Ubisoft GDC Europe 03 presentation);
    - Secret Level delivered a 829 build for PS2;
    - Long pre-production;
    - Short production.

From 829 to Splinter Cell: PLC

- Project Life Cycle – Postmortem:
  - We went fast through Prototyping & FP stages;
  - The team had no experience in FPS;
    - » Within a few month they became highly skilled;
    - » Would have been risky to develop from scratch.
  - The size of the XBOX team stayed relatively small.
  - During production, many risks were taken:
    - » Because we had a strong technology;
    - » Because the team was excellent.
  - PS2 and GameCube versions done successfully;
    - » Some optimizations could be done; ☺
    - » Ps2 Pre-production has been key.

UBISOFT

Strategic Highlights

Strategic Highlights

- Engine / Effort / Staff:
  - Middleware not used to reduce team size!
  - Really used as a way to:
    - – Put us at the same level as the competition;
    - – Let all the team focus heavily on the game features via:
      - » Implementing/tuning those features;
      - » Adapting the pipeline to our business goals;
      - » Refine the HP to help us bring the desired quality.
    - – Optimize the Project Life Cycles

Strategic Highlights

- Game Development Philosophy – part 1:
  - Middleware provides a big win in Prototyping…
    - » Use all appropriated middleware to focus on the development of the idea;
    - » Each prototype can be done with different Middleware (graphic style, animation style, behavior style, physics quality, sound quality, etc.)
  - … in Pre-Production phase to reach FP…
    - » At this stage, you know your game a little better;
    - » Choose a Game Engine to integrate all the prototypes;

Strategic Highlights

- Game Development Philosophy – part 2:
  - … in Pre-Production phase to reach FPP.
    - At this stage, you know your game & your constraints & the targeted HP:
      - » you start building your Production Pipeline and Game Engine;
      - » You can choose to build it from scratch!
    - You can choose an existing internal/external Middleware:
      - » The FP one;
      - » A better suited one;
  - Why is that an optimized Prototyping & Pre-Production??

Strategic Highlights

- Game Development Philosophy – part 3:
  - Proving ideas first!
    - » Reconsider them;
    - » Let them evolve.
  - Proving Production Pipeline and Game Engine second!
    - – Avoid building technology twice!
    - – Choose & schedule development according to key project parameters;
    - – Show:
      - » near definitive quality;
      - » How it respects budget/delays figures.

Strategic Highlights

- Game Development Philosophy – part 4:
  - In Production…
    - Production is often a race against time; to win you should:
      - » Have all your constraints understood by all;
      - » Be primarily optimizing the production pipeline to prepare the tools for the tuning phase;
    - Be optimizing the engine :
      - » To allow the design of special heavy scenes;
      - » To give more flexibility to artists;
      - » To allow last-minute features integration (competition);
      - » Prepare for the next title.
    - Reduce as much as possible other developments.

Strategic Highlights

- Game Development Philosophy – part 5:
  - In Production…
    - Synchronization:
      - » Stop synchronizing after alpha;
      - » After that, just get bug corrections.
  - While Tuning…
    - Prepare your pipeline to be highly flexible for tuning!
    - In a few days with focus groups, you can really improve a lot.
  - In QA…
    - Fast if only game specific debug is needed;
    - Painful if the Middleware was grabbed before a full testing;

Strategic Highlights

- Why building Middleware internally?
  - Not everything is supported:
    - » No GTA3 Game Engines available. ☺
  - Some features and/or technologies can be strategic for your company:
    - – Multiple genre of games, but common needs: Use a common SDK to:
      - » Share data better between projects;
      - » Favor one efficient process on all game projects.
    - – Avoid relying on disadvantageous business models:
      - » 1 choice, bad business model or licensing conditions;

Strategic Highlights

- Internal Middleware SDKs:
  - Online:
    - » Lighthouse 1.0 (2.0 in beta): Networking SDK
    - » GameService: Match making services
  - Animation: MORE
    - » Animation SDKs are often really basic;
  - Sound: DARE
    - » Sound is usually poorly considered by developers;
    - » the importance of tools is often neglected.
  - Menu: MAGMA
    - » For menu-heavy games (Rainbow Six3, etc.)

Strategic Highlights

- External Middleware:
  - Already in use:
    - Unreal2 – Epic;
    - Havok – Havok;
    - Karma – Criterion;
    - Renderware Graphics – Criterion;
    - Bink – RadGame Tools.
  - In a good way to be used:
    - Renderware studio – Criterion;
    - Unreal3 – Epic;
    - Half-Life2 – Valve;
    - Net-Z – Quazal;
    - Some others… ☺

UBISOFT™

Methodology

Methodology

- Hardware Projections – part 1:
  - Projections of the Data on the Hardware from an Engine perspective: The Hardware Projections.
    - » Understand how the Data behave on the original Hardware from the Engine perspective;
    - » Project on your Hardware to see what Data cost;
    - » Plan data & code adaptation according to the Hardware projection you target for each «map»(or even «sector»).

Methodology

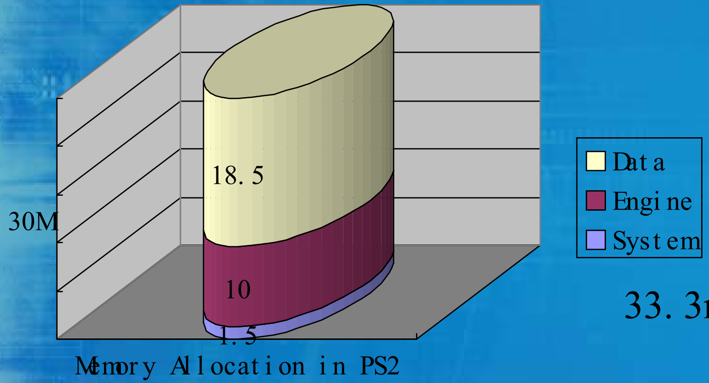
- Hardware Projections – part 2:
  - is intended to answer those questions:
    - » What are my engine constraints on each plate-form?
    - » What are the data configurations that make my game run correctly in N fps?
    - » How big can my levels be?
    - » How detailed can my levels be?
    - » How can I solve memory, CPU, GPU, issues?
- Tool for the programmers, managers and… artists!

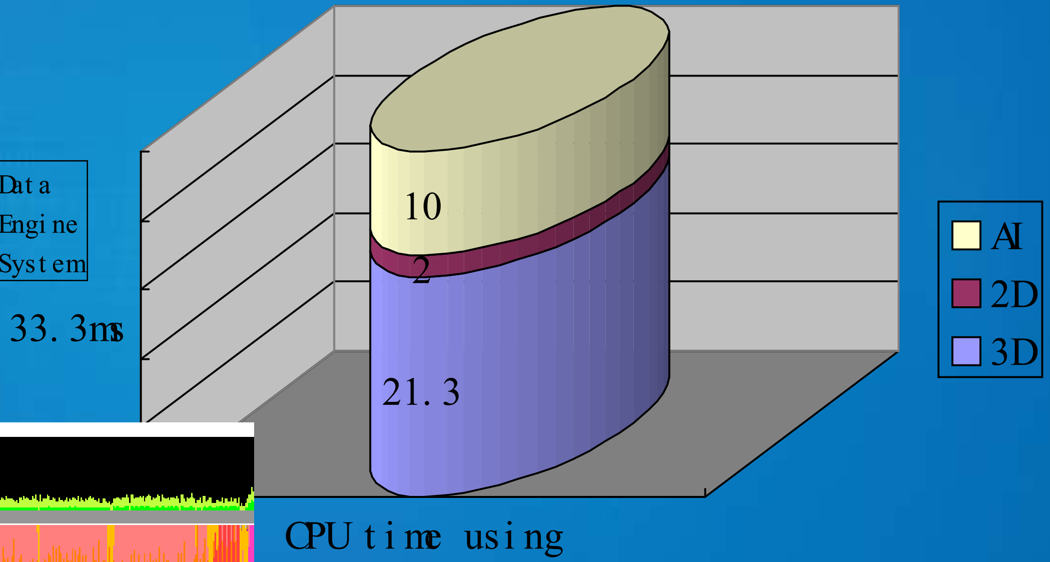
Methodology

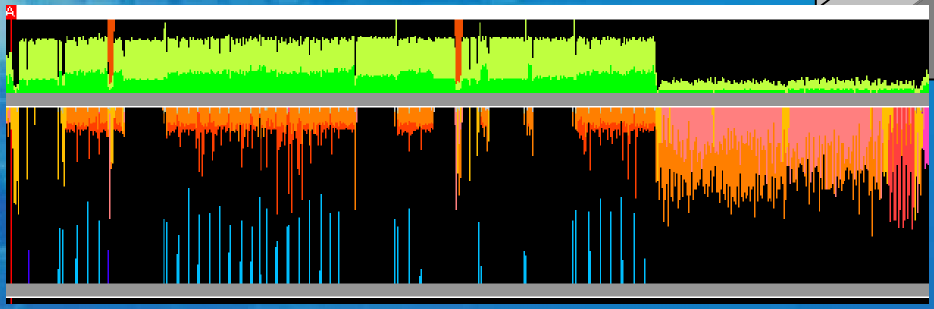
- Hardware Projections – part 3:
  - Theory:
    - » Engine usage is profiled on CPU, IOP, VU0, VU1, GS, RAM, VRAM from Benching Data sets. From all that, we get the detail for all type of data;
    - » Those Benchmark Data sets will be manipulated until we reach the limits of the engine on the hardware;
    - » This will set constraints on Level design (and feature usage).
  - In practice, the projection:
    - – Can be as detailed as needed;
    - – Can be done for each key feature.

Methodology

- Hardware Projections – part 4:







Methodology

- Tech Design: 3 steps
    - Prototyping
        - » Select a set of Game Engines and SDKs;
        - » Prototype all your ideas to test them.
    - FP
        - – Merge all prototypes in one Game Engine;
        - – Start studying the best suited Game Engines:
            - » HP, development & adaptation plans.
    - FPP
        - » Adapt the selected Game Engine to your needs;
        - » Port your FP version and enhance it;
        - » Create new FPP levels;
        - » Schedule production development.

UBISOFT

Methodology

- Documenting:
  - Useful for:
    - » new comers;
    - » people that reuse your technology.
  - When very few newcomers, let the people that reuse your techno make the documentation, because:
    - » It takes time to get right & update;
    - » You'll forget many « details ».

Methodology

- Closing Kit:
  - For each stage of a cross platform project, a closing kit is done;
  - Contains:
  - **1) The complete source code of the engine.**
  - **2) The complete source code of all tools.**
  - **3) The executable of the code.**
  - **4) The executable of all tools.**
  - **5) The complete raw data.**
  - **6) The complete Game data.**
  - **7) A "Workflow" document, containing:**
  - **a- The "step by step methodology" to recreate the build from the raw data & source code, especially how to use the tools.** (see template 1 & 2 below).
  - **b- All engine documentation / guidelines** (see template 3 below).
  - **c- A list of all types of data files**, their purpose, their structure (see template 4 below)
  - **d- An exhaustive list of all the internal tools involved in the Game Data creation** (+ detail their specific role & who uses it).
  - **8) A "Hardware configuration" document, listing <u>team by team</u> the system requirements necessary to develop.**
  - **9) A "Software configuration document", listing <u>team by team</u> the software necessary to develop.**

UBISOFT™

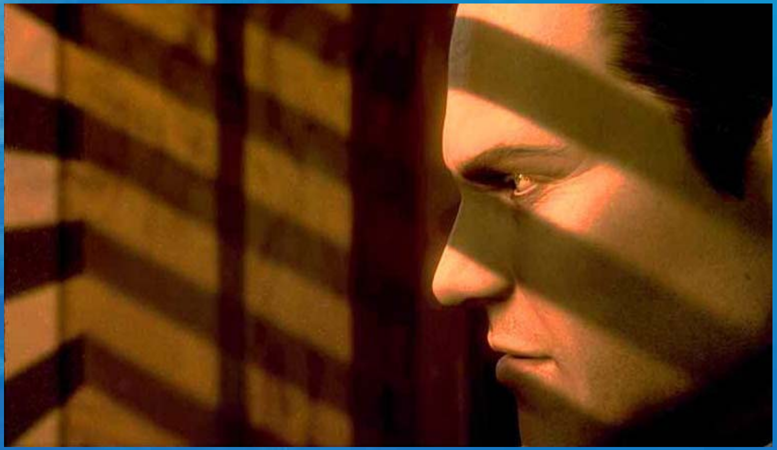
Conclusion

Conclusion

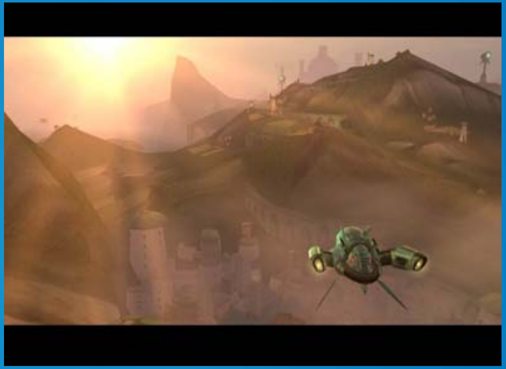
- Middleware:
  - Is key for an efficient pre-production:
    - » Diversity counts!
  - Helps you:
    - » get faster to the level of the competition;
    - » focusing on enhancing on that basis;
    - » focusing your team on game key features;
    - » focus on enhancing the production pipeline.

Conclusion

- Get some great inspiration
  - And focus on it!
    - » A solid technology base helps.
    - » Use Middleware to push the limits, not to reduce your staff!



Conclusion



Conclusion

- What's next on our side?
  - Internal SDKs:
    - » New SDKs dedicated to some specific common features/services across our games;
    - » Synchronization & evolution of existing ones.
  - Already working on Next Gen:
    - » Multi-Generation pipelines design;
    - » Single Next Generation pipeline design.

QUESTIONS?

Jmerceron@ubisoft.fr

Thanks