The Technology of

Uncharted: Drake’s Fortune

Naughty Dog

Christophe Balestra, Co-President
Pål-Kristian Engstad, Lead Programmer
Introduction

- Project Code Name: ‘BIG’
- 3 year project
- Pre-production: ~10 people for 12 months
- ~70 people + ~5 contractors
- 6 designers, 18 programmers, ~50 artists
Development

- Started the project with ZERO line of code
- Prototyped shaders on PC
- Animation was priority #1
- Over-designed our first set of tools
Tools

- Keep them SIMPLE!!!
- Few tools rather than a big über tool
- Expand your current set of tools
Cygwin

- Like a linux shell
- Helps controlling people’s environment
- All tools must be run from Cygwin
BAM

- Asset Manager
- Everything on the network
- Linux server - symlinks
- Check-out, Check-in but NO SYNC
GUI tool to describe what actors and levels are made of

All the data and change lists stored in MySQL: Bad idea!!

Switching to text files and Perforce
Very simple command line tools
BA builds actors
BL builds levels
Stats
Visibility

- During BL a PVS is pre-computed per level
- Rendering the level with OpenGL from sample points
- Computed once a day and re-used cache
- Split boxes
- Can easily add sample points
Distributed build

- Using Naughty Dog Distributed System
- Very simple
- Runs command line tools
- Everything stored in MySQL
Shaders

- Über Shader file
- `#define = feature`
- Not great for dependencies
- Compiled during BA/BL
- Each actor or level contains its shader code
Material Editor

- GUI tool connected to Maya to create shaders
- Artists select an .fx file and choose features
- Artists loved it but ...
- Not very stable, data & change list stored in MySQL
- Working on connecting it with the game
Charter

- GUI tool to create game-play contents
- Maya too slow
- Regions, Nav-Meshes, Spawners, Cover points...
- Quick iteration time
DC

- Data Compiler
- LISP based
- Original design was to create data structures
- Added scripting
- Render settings, AI, animation, region scripts
- Realtime linking
Static Lighting

- Global illumination
- Direct color, indirect color and direction per vertex
- Light probes for objects
- Too slow, couldn’t distribute
TAME

- GUI tool to manage text localization
- HELPED US A LOT!!!
- Producers in Europe and Japan can manage everything without us
- MySQL
<table>
<thead>
<tr>
<th>Name Id</th>
<th>Text</th>
<th>Last Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>00A_01A_SEQUENCE_01</td>
<td>I'm here off the coast of Panama, where we've just recovered what we believe</td>
<td>2007-06-05 11:47:29</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_02</td>
<td>to be the coffin of legendary explorer Sir Francis Drake, who was buried at sea over 400 years ago.</td>
<td>2007-08-06 17:09:31</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_03</td>
<td>Are you sure you wanna be defiling your ancestor's remains like that?</td>
<td>2007-08-06 17:10:36</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_04</td>
<td>You make it sound so dirty. (laughs) Besides, I thought you didn't believe me.</td>
<td>2007-08-06 17:10:50</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_05</td>
<td>Well, I did do my research. And apparently, Francis Drake didn't have any children.</td>
<td>2007-08-06 17:11:25</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_06</td>
<td>Well, history can be wrong, you know. For example - you can't defile an empty coffin.</td>
<td>2007-08-06 17:12:10</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_07</td>
<td>What the hell? (laughs) You devil.</td>
<td>2007-08-06 17:12:37</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_08</td>
<td>What is it? C'mon, hold it up - No, no, no - no way.</td>
<td>2007-08-06 17:12:57</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_09</td>
<td>The deal was for a coffin, that's it. Wait a minute, if my show</td>
<td>2007-08-06 17:13:26</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_10</td>
<td>-hadn't've funded this expedition, you wouldn't've - (\text{Hey, hey...})</td>
<td>2007-10-08 00:38:11</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_11</td>
<td>You got your story, lady.</td>
<td>2007-08-06 17:14:21</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_12</td>
<td>-Look, Mr. Drake, you signed a contract. (\text{{laughs}})</td>
<td>2007-08-06 17:14:40</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_13</td>
<td>-I have a right to see every single thing that... (\text{-Whoa whoa...})</td>
<td>2007-08-06 17:15:25</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_14</td>
<td>Could you hold that thought?</td>
<td>2007-08-06 17:16:35</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_15</td>
<td>Sully? Uh, we got some trouble. Hurry it up.</td>
<td>2007-08-06 17:18:12</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_16</td>
<td>Okay, okay - what's going on? Uh... Pirates.</td>
<td>2007-08-06 17:18:31</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_17</td>
<td>Pirates?!</td>
<td>2007-08-06 17:18:51</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_18</td>
<td>Yeah, the modern kind. They don't take prisoners. Least not male prisoners.</td>
<td>2007-08-06 17:20:04</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_19</td>
<td>Wait, what are you talking about?</td>
<td>2007-08-06 17:20:33</td>
</tr>
<tr>
<td>00A_01A_SEQUENCE_20</td>
<td>Uh, sh- shouldn't we call the authorities or something?</td>
<td>2007-08-06 17:20:50</td>
</tr>
</tbody>
</table>
Fileservers

- We love Linux
- Linux is fast
- The game connects to our own file system running under Linux using the debug port
- 80 MB/sec
Programmer Pipeline

- Linux again!!
- 1 linux box for every 2 programmers
- Putty
- It’s FAST!!!
- Multithreaded build
- SN Debugger & Gcm Replay
Artist Pipeline

- Maya & ZBrush
- Material Editor
- BuildBig
- BA/BL
Designer Pipeline

- Charter
- BuildBig
- BA/BL
- DC
All IO done with FIOS

Everything is compressed using Edge

We stream a lot and all the time: levels, actors, sounds, music, textures

Pre-cache
Loading Scheme

- Levels and actors are .pak files
- .pak file is made of pages
- Heap of 172 pages
- No fragmentation
Texture Streaming

- Having a hard-drive on every PS3 is HUGE
- Very easy to do ... 3 days of work
- Adds a lot to the quality of your game
- Everyone must stream textures!
- Defragment memory every frame
Texture Defragmentation

Step 1

Step 2

Step 3

Step 4

Step 5
Collisions

- Sphere, Capsule, Convex Polytope
- Concave Geometry, kd-tree Polygon soup
- SPU elf per Shape
- 32 Objects per Batch
- ~500 asynchronous ray-cast every frame
Physics

- World split into Islands
- One SPU job (solver) per island
- Successive Relaxation LCP solver
- General constraint system: rag-doll
- Constraints & limits derived from “range-of-motion” animation
**SPUs**

- Scene traversal
- Geometry processing
- Spherical Harmonics to cube maps
- Particle simulation

**Water**

- Animations
- Decompression
- Collisions
- Physics
- Path Finding
Animation

- Using SPUs to decompress and blend animation tree
- Blend tree up to 25 animations (Drake)
- Use DC to describe animation states
- Layered animation system
Scene Traversal

- Visibility frustum culling
- PVS lookup
- Sorting
- Render set up
- Mesh processing set up
Scene Traversal

- Visibility frustum culling - SPU
- PVS lookup - SPU
- Sorting - PPU
- Render set up - PPU
- Mesh processing set up - SPU
Mesh Processing

- Use SPUs to offload RSX
- Decompression
- Skinning
- Back-face culling
- Also used to collide with rendered geometry: decals, IK, some gameplay collisions
Scene Rendering

- Shadows
- Dynamic Lighting
- Opaque Geometry
- Alpha Blend Geometry
- Post Processing effects
Sun Light Shadow

- Tried a lot of solutions: problems!
- Simple idea by Guerilla (Killzone)
- Reduce flickering: fixed world space sample points
- SSM: orthographic shadow map
- Cascaded shadow maps
Cascaded Shadow Maps
Shadow Blockers -> Sunlight Shadow Pass

Opaque Geometry -> Depth Only Pass

Sunlight Shadow Pass -> Shadow Depth Buffer
1216x1216 : Z16

Depth Only Pass -> Depth Buffer
2x1280x720 : Z24
Dynamic Lighting

- Render opaque dynamically lit geometry: world normal + specular exponent in screen space
- Divide the screen into a grid
- Find which lights intersect each cell
- Render quads over each cell calculating up to 8 lights per pass: results in a light buffer
Opaque Lit Geometry

Get World Space Normal and Spec Exp

Normal Buffer
2x1280x720 : RGBA8
Dynamic Lighting
Opaque Rendering

- Shadowing and lighting already done
- Material processing + direct lighting
- Output is HDR (logluv)
- Pixel shader bound: vertex processing mostly done on SPUs
- Dithering out some geometry
Logluv Buffer
2x1280x720: RGBA8

Depth Buffer
2x1280x720: Z24

Resolve color + Fog

Color Buffer
1280x720: 4xFP16
w = dist from camera

1x Depth Buffer
1280x720: Z24
Alpha-blended Geometry

- Rendered to FP16 1x buffer
- Water
- Particles
- Glass
Water Meshes

Shadow Depth Buffer
1216x1216 : Z16

Color Buffer
1280x720 : 4xFP16
w = dist from camera

Logluv Buffer
2x1280x720 : RGBA8

Color Buffer
1280x720 : 4xFP16
w = dist from camera

1x Depth Buffer
1280x720 : Z24

Water Pass
Particles

- Simulation and rendering set up done on SPUs
- Wrote specific shaders
- Switch to down sample buffer on frame rate spike
Post Effects

- Depth of field
- Blur
- Tonemap
- Bloom
- Saturate
- Tint
- Distortion
- Motion Blur
- Blend
KEEP IT SIMPLE!
WE’RE HIRING!!!!!!

if you’re extremely talented ...

candace_walker@naughtydog.com
Thank you

christophe_balestra@naughtydog.com
pal_engstad@naughtydog.com