Shipping Sims 2

Andrew Willmott, MAXIS

Why is Sims 2 Interesting?

- *Massive amounts of content*
  - *Animation heavy*
  - *Sound heavy*
- *Massive amounts of people!*
- *Graphics: "The users design the levels"*
- *Visual game scripting language (Edith) drives much of gameplay.*

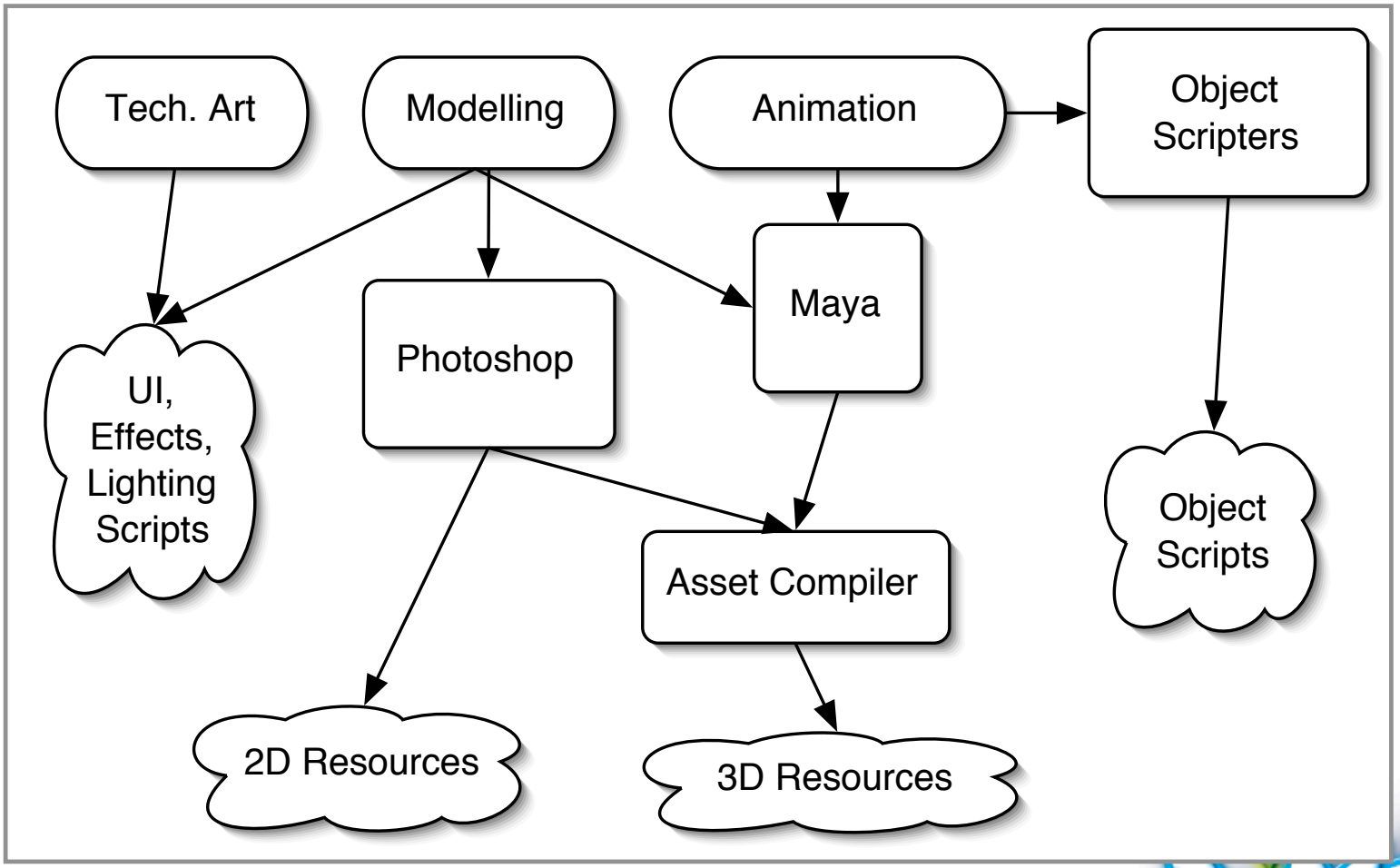
Long Project

- *Started in late 2000*
- *Mostly a small research team (5–20 people) for a few years*
- *In full production for 2+ years*
- *250+ people at the end*
- *Slipped! From January ‘04 to September ‘04*
- *Extended crunch*

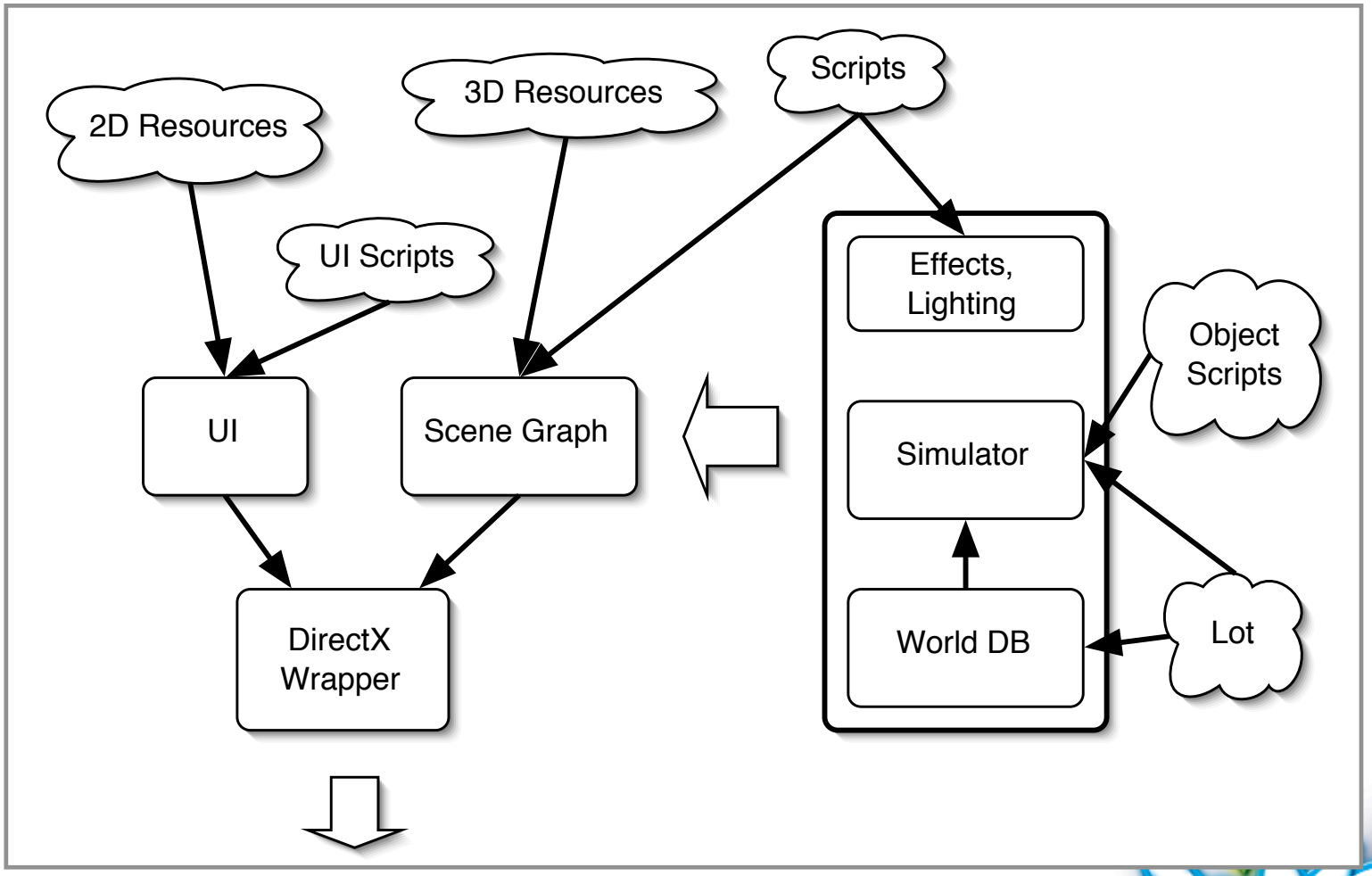
Stats: Highlights

- *4 CDs*
- *11,000* ***shipped*** *animations*
- *1.1 million lines of code*
- *2,400 UI images*
- *1* ***GB*** *sound data*
- *8 GB development build footprint*

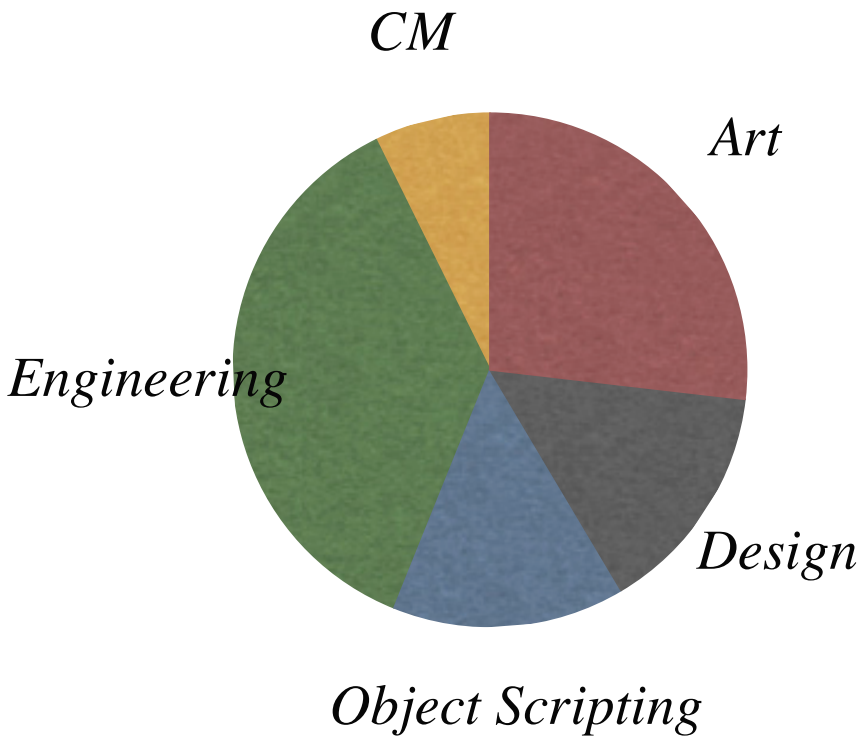
Sausage Factory: Content



Sausage Factory: Code



Areas We'll Look At



- *Lessons learnt, moving forward*

Art

Stats: Art

- *11,775* ***shipped*** *animations*
- *4,500 models, 8,100 textures*
- *50,000 lines of effect scripts: 2000 total effects*
- *57 Movies*
- *3.5GB of source data for Sims, 630 MB for objects*

Stats: UI

Art

- *2,400 UI images*
- *240 UI scripts*
- *942 catalog items*
- *21 language string packages*
- *92 cursors*

Art Staffing

- *About 35 artists at peak*
  - *17 Animators (12-14 by ship)*
  - *13-14 Modellers*
  - *3 Technical Artists*
  - *2-3 UI Artists*
  - *2-4 Effects Artists*
- *Some overlap*

Art Tools

- *Photoshop, Maya-based from the beginning*
- *Brought over all Sims 1 animations, targetting new skeleton*
- *Used none!*
- *MEL front end for material system*
  - *In general, MEL very useful for adding UI, absorbing pipeline logic*

Art Pipeline

- *Photoshop -> TGA -> Asset Compiler (Go2SCO)*
  - *added direct PSD support: useful because of **layers***
- *Maya -> EA3 -> Asset Compiler*
- *Artists check in source files to perforce*
  - *Build machine runs them through pipeline, sends error summary to submitter (a few minutes)*
  - *Also updates animation/models report: web-based view of every asset and its stats*

The Skeleton

Art

- *27 facial targets packed into 4 delta streams*
- *64 weighted bones*
- *116 bones total, including grips and other registration points*
- *Skeleton not locked*

Modelling

- *Lucky enough to have continuity from Sims 1x*
- *Sims: maintained single texture sheet approach, but composited tops/bottoms together*
- *Artists weren't given a poly budget: batch count is all*
- *Ditto textures(!)*
- *Contracted out object LODs: tried to always have a single-material LOD*

Modelling Feedback

Art

- *Had Texel Density visualisation mode, for identifying "hot" objects where texels were wasted*
- *Had an animated model/texture map display*
- *Could switch between four different shader paths in viewer to verify lighting was correct.*

Lighting

Art

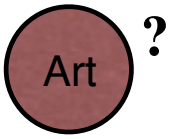
- *Art produced original lighting design. Had a TDD, and was prototyped.*
- *Two dedicated people tuning lighting system via script system*
  - *System evolved according to art, especially early on*
  - *Artificial contrast boosters, emphasis on gradients*
  - *Downside: a large number of knobs, confusing for someone new.*

Level of Detail

Art

- *Static LOD*
- *Never formally spec'd*
- *All Sims had one LOD (two models total), about 50% of objects had at least one LOD.*
- *Initial overly-ambitious plans left us scrambling at the end*
- *Dropped dynamic LOD switching due to visual glitches*

Sound



- *143 MP3s*
- *43,000 spx vox samples!*
- *2,600 footsteps alone + 7,500 ambient samples*
- *16,000 unique sound events*
- *Fully half our data footprint:* ***1 GB***
- *35,000 sound resources loaded going into the neighbourhood*

Design

Dsn.

Design

Dsn.

- *Difficult!*
  - *Need to improve on one of the best games of all time*
  - *Attract new players*
  - *Without putting off previous players*
- *Diverse audience*
  - *Past surveys have shown even split between at least four different styles of play*
- *Aiming for 90+ metacritic rating*

It Gets Worse

Dsn.

- ***Big*** *pressure from EA! Constant demo pressure worked against design gelling*
- *How much new stuff is enough?*
- *The 'spiralling delay' trap–longer it takes, better it has to be*
- *Some of earliest ideas cannibalized for 1x expansions, plus people who originated them*
- *Changed one of the lead designers in 2003*

New Gameplay

Dsn.

- *Movies*
- *Aging!*
- *Generations and Genetics!*
- *Big Life Moments & Cinematics*
- *Aspirations?*
- *Wants & Fears?!*
- ***Done**. Phew.*

Process

Dsn.

- *After slip, needed extremely tight control to ensure we hit our mark:*
  - *Change review—any new design feature thoroughly vetted, most dropped*
  - *Feature producers dedicated to seeing a particular feature completed*
  - *Big new areas were tightly compartmentalized*
  - *E.g., wants and fears: two engineers working closely with OE and design/production*

It shipped!

Dsn.

- *The delay was crucial*
- *Finally got enough compelling new gameplay to bring back previous buyers.*
- *Finished (most of) engineering*
- *1 million units sold in 10 days*
- *Sales in Europe > North America*
- *Lesson: Never give up!*

But at a Cost

Dsn.

- *Constant design change was a negative for the rest of the team*
- *Most of Maxis moved to EARS in early 2004*
  - *Corresponding loss of studio identity*
- *Various talented people burnt out, some leaving*
  - *Worry that will impact future hiring prospects*
- *Big incentive to learn how to manage "big product" process better!*

Object Engineering

- *18-19 OEs*
- *1,700 game objects!*
- *Simulator and object scripts drive all gameplay*
- *Mostly nice coupling of logic and associated game object*
- *Simulator closely coupled with C++ primitives*
- *Sometimes mismatch between complexity of C++ primitives and simplicity of Edith*

Artist/OE Interaction

- *Essentially, artists provided source animation, OEs supplied blending (not ideal)*
- *A big problem synchronising*
- *Skunkworks project produced "Clockwork", which allowed easy previewing of animations and associated effects*
- *OEs could use this to explore art assets when writing scripts, rather than bugging artists*

Edith

- *Less interesting than you think!*
- *Actually, visual scripting doesn't work very well*
  - *No revision history*
  - *No good search/replace*
  - *Single edit: once person at once*
  - *Difficult for script sweeps*
- *At the end of the project, all OEs wanted to move away from it*

Edith

- *Positives: having a good debugger is crucial*
- *After SC4 and Sims 2, studio consensus is that Edith has more cons than pros*
- *The future: Lua*
  - *Used on SC4 with some success*
  - *In use on various next gen titles*
  - *But debugger a work in progress*

Engineering

- *Around 28 engineers at peak. Very roughly:*
  - *4 Simulation*
  - *5 UI*
  - *4 Graphics only, 7 Graphics/Gameplay/World*
  - *4 General*
  - *2 Animation*
  - *1 Audio*

Stats: Engineering

- *1.1 million non-comment, non-blank lines of code*
  - *325,000: framework code shared across the studio*
  - *80,000 graphics engine, 45,000 animation*
  - *110,000: Shared between app/tools*
  - *540,000: game-specific. Gameplay, UI, world construction, lighting...*
- *17,000 lines of material/shader scripts*
- *1,000+ lines camera/catalog/lighting*

Building Stuff

- *Dedicated engineer: The World DB*
  - *Terrain, all house geometry, object location*
- *Bridge between gameplay and engine*
- *Kept tile-based system*
  - *Mostly for UI/gameplay reasons*
  - *Actual world DB code mostly only cared about walls and rooms as quad-edge data structure*
  - *But this wound up being overly general*

Routing

Eng.

Routing

Eng.

- *Achilles heel of Sims 1*
  - *Painful: "Party syndrome"*
- *Contracted company to write a replacement*
  - *But, slow, memory hungry, not that good*
  - *Integral part of gameplay: needs **iteration***
- *Instead, dedicated an engineer to the system*
- *Worked with OEs to solve most problems*

Routing

Eng.

- *Standard, quad-tree based*
- *Smarts improved, higher tile granularity*
- *Tied to simulator: gardener, ghost, fireman*
- *"Step over"*
  - *Essential for all those messes on the floor*
- *"Side stepping"*
  - *Two Sims can pass each other in a narrow space*

Animation

Eng.

- *Full multi-channel blending, two-bone IK*
- *Look at*
  - *Sims can glance at each other on room entry etc.*
- *Hair bounce*
- *Standard Reach*
  - *Used for Sims to hit various targets*
- *Cinematics*

Effects System

Eng.

- *Script-based system*
  - *Effects composed from "components": particle systems, decals, sounds, models...*
  - *Hierarchical: can nest effects, "meta" particles*
  - *Random walks, particle stretch, attractors, colliders*
- *Key: all scripts are **hot-loaded**, rapid iteration*
- *Handles UI too: thought balloons, most build mode tools*

Example: Fishies

Eng.

- *The fish tank is all an effect*
  - *Fish are model particles with random walks bounded by colliders*
  - *Game can kick an effect between states*
  - *When fish die, wind force floats them upwards until they hit tank's "top" collider*
  - *On collision, die and spawn dead fish model*
  - *Can also switch to state with food attractor*

Neighbourhood

Eng.

- *Lots were imposterized on lot exit*
  - *All walls and floors captured into small set of texture pages via render targets*
  - *Object substitution for "important" objects*
- *SimCity tie-in with terrain generation*
  - *Roads and trees imported directly*
  - *Everything else is effects*
  - *User can even place these*

Lighting System

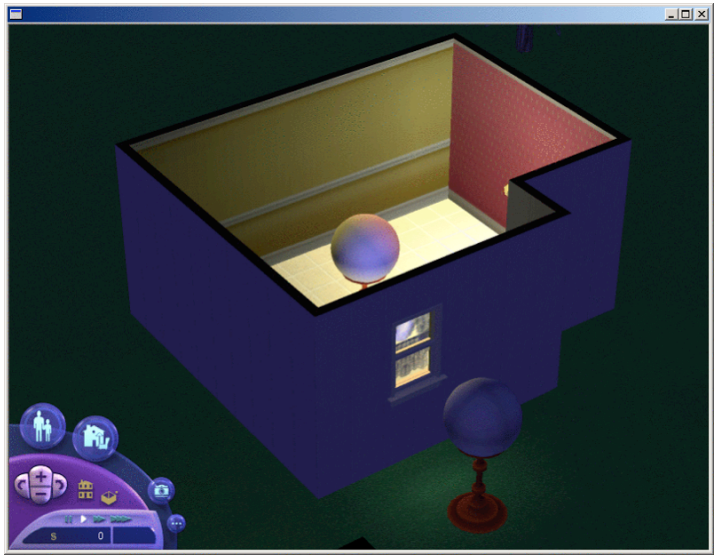
Eng.

- *Lighting was room based*
  - *Each room had a light rig generated for it automatically*
  - *User-placed lights only affected objects in that room*
  - *Portals transmitted light between rooms*
- *Time of day states, smooth transitions*
  - *But, states cut to day/night only. Smooth object light transitions killed due to engine issues.*

Lighting

Eng.

Portals



Room Light Rig

Lighting

Eng.

- *Exterior lighting predefined*
- *Objects and portals have various light multipliers depending on inside/outside*
  - *Falloff, cutoff, intensity, directionality, etc.*
  - *Had 2x "over-bright" lighting*
- *Floor and walls were light mapped*
  - *2D Diffusion algorithm used for "faux" radiosity*

Shadows

Eng.

- *Terrain and house: height map shadows*
  - *Fast CPU-side algorithm*
  - *Baked directly into light maps*
  - *Allowed simple object-as-a-whole shadowing*
- *"Cookie-cutter" projective shadows*
  - *Dynamic only for Sims, and some animating objects*
  - *Static for objects, updated in a staggered manner with a frame budget.*

Shadows

Eng.

- *Tricks*
  - *Blur and threshold, so could use pretty low-res maps*
  - *Share for identical objects with same rotation*
  - *Many shadows packed into a single render target*
- *Indoor: GUOBS*
  - *Prebaked "straight down + ambient" shadows*
  - *Contact shadows, e.g. for wall objects*

Scene Graph

Eng.

- *Graphics engine was a fully general scene graph*
- *All model, camera, and hardware light manipulations were carried out via graph node manipulations*
- *A model was just a (tagged) subbranch of the scene*
  - *Could be many nodes: hundreds for a Sim*
  - *Many operations involved traversing a branch*

Graphics Performance

Eng.

- *Many objects in a house*
- *Terrain and house split into sectors for culling and dynamic lighting*
- *As batch count hits the thousands, start to get CPU hit*
  - *Generic scene-graph-based graphics engine rebuilt display list every frame: CPU hit per part*
  - *DIP cost becomes prohibitive*

Solution: Dirty Rects

Eng.

Dirty Rects

Eng.

- *No, really. (And we thought SC4 was it.)*
- *Initial "hold" scheme gradually morphed into an SC4-style static/dynamic layer model*
- *Cause of some of our card compatibility problems. (Copying depth surfaces is tricky)*
- *Lighting system required complicated last-minute update to generate dirtied areas: tile-based*
- *Shadows, particle systems, etc. retrofitted*

Target Platforms

Eng.

- *Order of magnitude differences between our low end and high end in many categories*
  - *Memory, VRAM, Card capabilities*
- *Had to support non-T&L commodity Intel hardware*
- *Pixomatic fallback for unsupported cards*
- *Biggest target was DX7-level cards*

Game Configuration

- *Complicated!*
- *Used SC4-derived configuration system, but with more logic in the scripts*
- *Cards are identified from vendor ID, plus driver version*
- *Special cases as appropriate*
- *Usual headache estimating texture memory*
- *Relying on caps bits does not work* ***at all***

Memory

Eng.

- *A lot of STL*
  - *Not always efficient, but golden for leak prevention*
- *Ref counting and interfaces: AutoRefCount<>*
- *Custom allocators*
  - *Per-object pools: very low allocate/dealloc overhead*
  - *A refined cross-platform evolution of dlmalloc*
- *Per-class leak detectors*

Leak Observations

- *Ranking leak causes:*
  - *By far, manual news/deletes*
  - *Then manual refcounting.*
  - *Finally, ref-count loops due to improper Init()/ Shutdown().*
- *Biggest finalling leak:*
  - *SC4: Lua!*
  - *Sims 2: Logging system!*

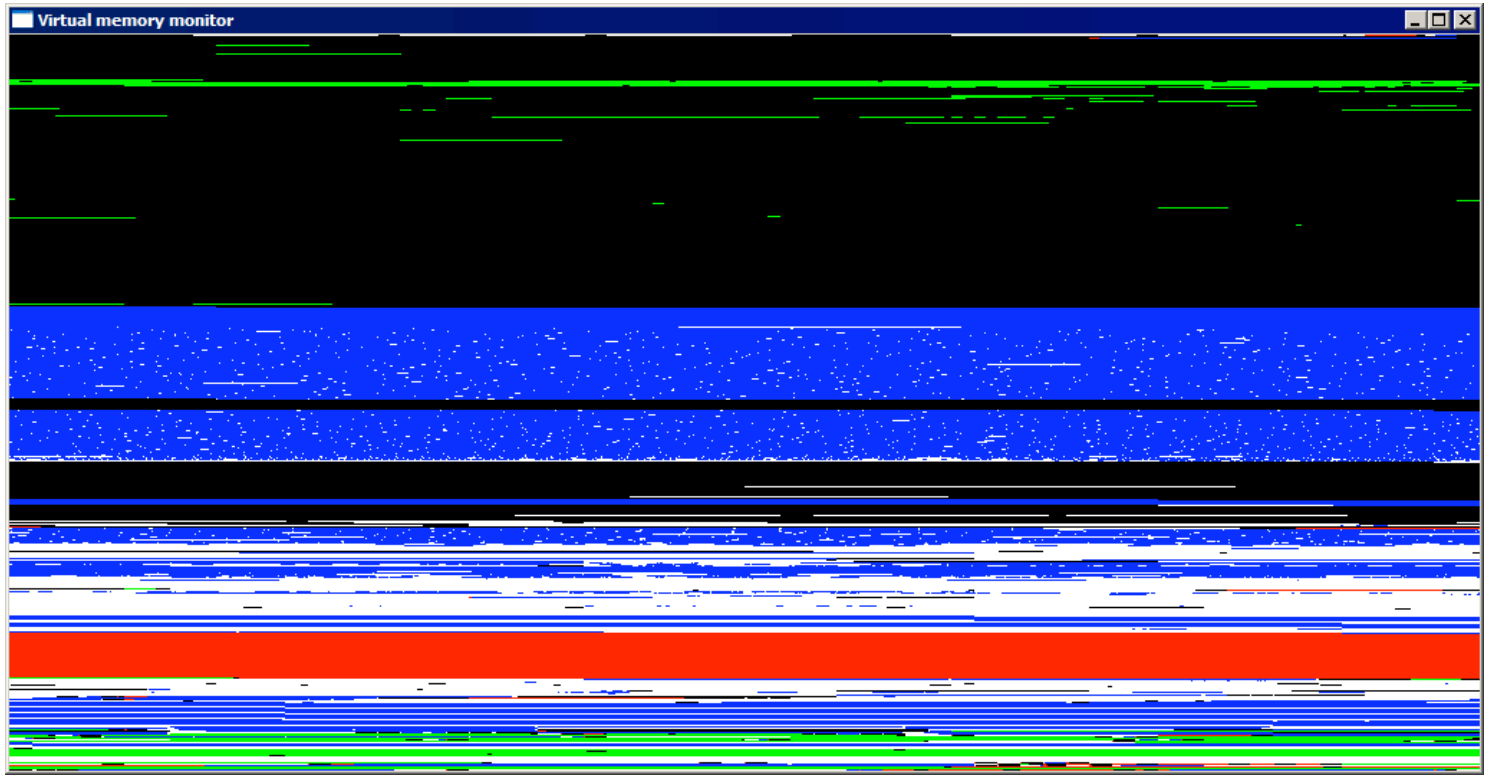
Virtual Memory

Eng.

- *Traditionally the crutch of PC games*
- *Free lunch is running out: Virtual Memory fragmentation and exhaustion rearing its head*
- *DLLs carve off large amounts of address space*
- *Operating system takes ever larger amounts*
- *Memory fragmentation can bloat application's footprint*

Virtual Memory

Eng.



2GB

Blue: reserved, white: committed, green: exe/dll, red: mmap

Resource Management

Eng.

- *Key based: originally 96 bit keys*
  - *Resource UID/Group UID/Instance UID triplet*
- *Unpacked form: resource is a file*
- *Packed form: package compiler mapped resource directory hierarchy into a set of large files*
- *Worked well for previous large-content games*

Problems

Eng.

- *Overused resources. Some simulator resources were tiny: 12 bytes each!*
- *Models stored as scene graph nodes etc.: a single model could easily become 30-50 resources*
- *15,000 models/anims -> 100,000+ resources*
- *Key collision: Sims 2 engine allocated resource/ group via class UIDs, and hashed string file names into instance ID. A hash is* ***not*** *a UID.*

Problems

Eng.

- *What about custom content?*
  - *Player A loads skins created by players B & C*
  - *Naming doesn't work: they both called it "Bob"*
- *Large number of files meant the development build's load time became prohibitively slow*
  - *Load logs were main tool for identifying problem areas: simple time-stamped checkpoints.*
  - *Added development build caches*

Solutions

Eng.

- *In the end, just extended instance ID to 64 bits*
  - *Case study in making risky changes late in development. (Happened **after** BodyShop ship)*
  - *Work was done in a sandbox separate from main development line, and tested thoroughly before merge*
- *For custom content, relied on trusty 128-bit GUID*
  - *Alternative: use hashing, deal with collisions. But gets complicated fast. Simple brute force solution is preferable.*

Configuration Management

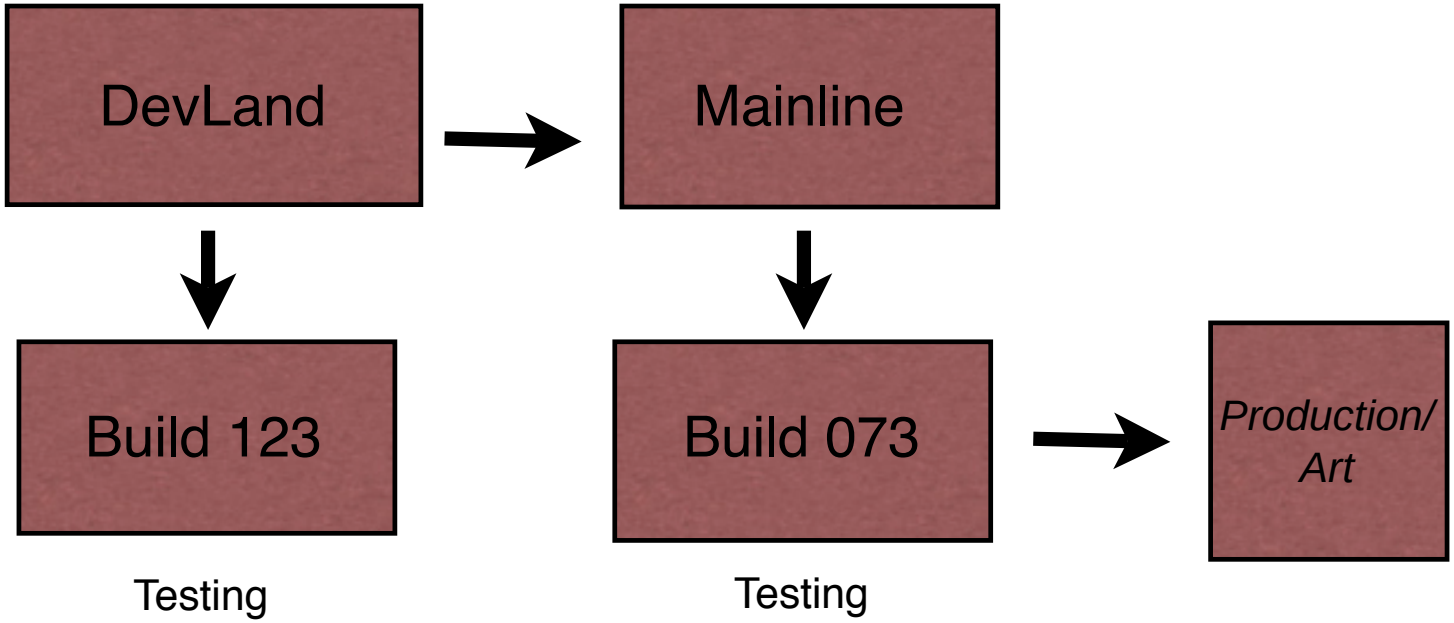
- *Soaked up a lot of effort: 6-8 engineers*
- *Testing*
  - *Drove the game through the command console*
  - *Tests could then be scripted using a simple command script. 285+ test scripts!*
  - *Highly successful approach, used on a number of previous products*
- *Simple "sniff" test required before all check-ins.*

Configuration Management™

- *Adopted "DevTrack" bug database during SC4*
  - *Awful—slow and buggy, but stuck with it, improved*
  - *Interaction with testers limited to this*
  - *Testers: hundreds*
- *"Robbie" tool for rolling out builds*
  - *Builds copied incrementally. Could be slow*
  - *No rollback facility!*
- *Stack traces to web site*

Source Control

CM



- *Two code lines, builds from both were tested*
- *Pre-checkin code reviews tried early and abandoned. Reinstated during finalling.*

Lessons Learnt

What Went Right

- *We could always hit art lock. The explosion in art assets required turned out to be the least of our scaling problems*
  - *Learn from the animation industry—they've been doing this for a while!*
  - *Hire from the animation industry. As content gets larger, processes get ever more similar*

Art Lessons

- *Need stable design!*
  - *Too many cooks syndrome*
  - *Need concept art, templates, established look*
- *Modellers <-> animators <-> OEs sit closer together*
- *More consultation from engineering. Tighter turnaround for code changes*
- *Lack of technical art types hurt*

Design Risk

- *Well known: cost of a bug increases the later you catch it*
- *Corollary: cost of design changes increases **exponentially** the later they occur*
- *But Sims 2 couldn't afford to ship without getting the design right*
- *Caught between a rock and a hard place, but need to avoid this in future.*

Engineering Lessons

- *Concentrate on the game, not the engine*
- *Beware of cathedral building: Get systems in place early*
- *Scene graph belongs in the pipeline*
- *Never ignore value of shipping-hardened code*
- *Don't contract out core gameplay components*
- *More urgency!*

What Went Wrong

Transporter Accident #231

Over-Engineering

- *Some areas of the code base were massively overengineered*
  - *E.g., pixel formats were represented by a number of COM interfaces. 2200+ lines of code in all.*
- *Bubble-wrap syndrome*
  - *Feels like it should be simpler to do X!*
- *Prefer toolkit to one-stop-shop*
  - *Too generic = too hard to change and iterate*

Template Meta Programming

- *Sims 2 Math/Vector library used this*
  - *Performance improvement was never actually measured by writer. Turned out to be a slight* ***decrease*** *in performance in optimized release build*
- *Negatives:*
  - *Impact on debug speed was* ***horrific*** *(75% hit)*
  - *Very difficult to read*
  - *Very difficult to debug: deep stack traces*

API Churn

- *Lock low level parts of the game well before the final phase of development!*
- *API churn in low level systems is unacceptable*
  - *Engineers can't keep track of current feature set, or propagate knowledge about that feature set*
  - *Introduction of subtle bugs*
  - *Can't build on sand*

Engineering

- *Main problem of engineering team: lack of productivity for some core tasks*
  - *Spent weeks or months trying to do some things "the right way"*
  - *Planned overly-ambitious systems, ran out of time to implement them: Cathedral building*
- *But... what's wrong with taking time to build the cleanest and most generic system possible?*

Opportunity Cost Examples

- *We have nice normal maps. They only show up on a 128MB+ card*
- *Static LODs: picked depending on machine type, don't change in the lot*
- *Shader path that consumed most dev. time (DX8) was dropped in last weeks*
- *Hacky game-side culling. (Objects hidden manually by world DB code.)*

Going Forward

- *How are we applying these lessons?*
- *Real Estate has location, we have…*

Preproduction

- *One of the biggest learning experiences of Sims 2: this is crucial for large projects*
  - *Explore and solidify design with prototyping*
  - *Assemble look bibles, concept art, storyboards*
  - *Explore any new technologies necessary*
- *Then, slowly ramp up to full production*
  - *Must be sure to have all ducks in a row, and only add people when underlying systems are ready*

Why not before?

- *Sims 2 was small, "under the radar" research team for a few years*
- *Flipped directly to production when studio focus changed*
- *Maxis did not have a lot of experience with preproduction concept*
- *Deadlines and team sizes hadn't been such that it was crucial*

Communication

- *Return of the King used "pod" style of working*
  - *Small, tightly coupled, interdisciplinary groups*
- *Model has worked well at Maxis in an ad hoc way, now being adopted more formally*
- *Goal is to increase communication bandwidth where it matters,*
- *Also: flatten hierarchy*

Example: Pools

- *Cut for original January deadline*

Swimming Pools

- *Skunkworks team got together to save them:*
  - *Simple set of animations*
  - *Changes to router and world to treat as special room*
  - *Basic interacting water surface with caustics*
- *All put together in only a few weeks outside normal tasks*
- *Slip allowed more animations and lighting refinement*

Art

- *Already implementing pre-production in a number of development titles*
  - *Successfully using concept artist to rapidly explore both look and design space*
- *Work to have content validation tools working in place before production*
- *Replicate Sims 2 auto-content-build and content browser on other projects*

Engineering

KISS

- *It's all about managing complexity*
- *Prefer smaller, tighter teams focused on particular features*
- *Prefer rapid development when the new system is an unknown*

Technology

- *Transitioning to Renderware*
  - *Toolkit approach to graphics API*
- *Adopting effects system as shared technology*
  - *Also being used for rapid prototyping*
- *Switching to text-based scripting*
- *Better and more integrated game object/asset databases*
- *Continue to evolve scripted testing*

Acknowledgements

- *Leo Hourvitz*
- *Ben Thompson*
- *Paul Boyle*
- *Alec Miller*
- *Justin Graham*
- *David Benson*