

Additional Topics:

Computing Principles and Practice  
of a  
**Blockbuster Video Game**

Kenny Mitchell  
Research Lead  
Core Technology Group  
**Black Rock Studio**  
The Walt Disney Company

# Split/Second: Velocity

Released May 2010




# Video Game Industry

- Digital Economy
  - Creative Industries
    - Growth
- Retail Delivery
  - Developing online
    - Convenience
- Blockbusters
  - Modern Warfare £111m US
  - Grand Theft Auto IV £108m GB (2008)
  - FIFA £58m CAN
  - Wii Fit £45m JPN
  - Assassin's Creed £36m CAN
  - Need for Speed £18m GB
  - Batman £15m GB ([2009/2010 UK sales source Develop 100](#))



[gamedevmap.com](http://gamedevmap.com)

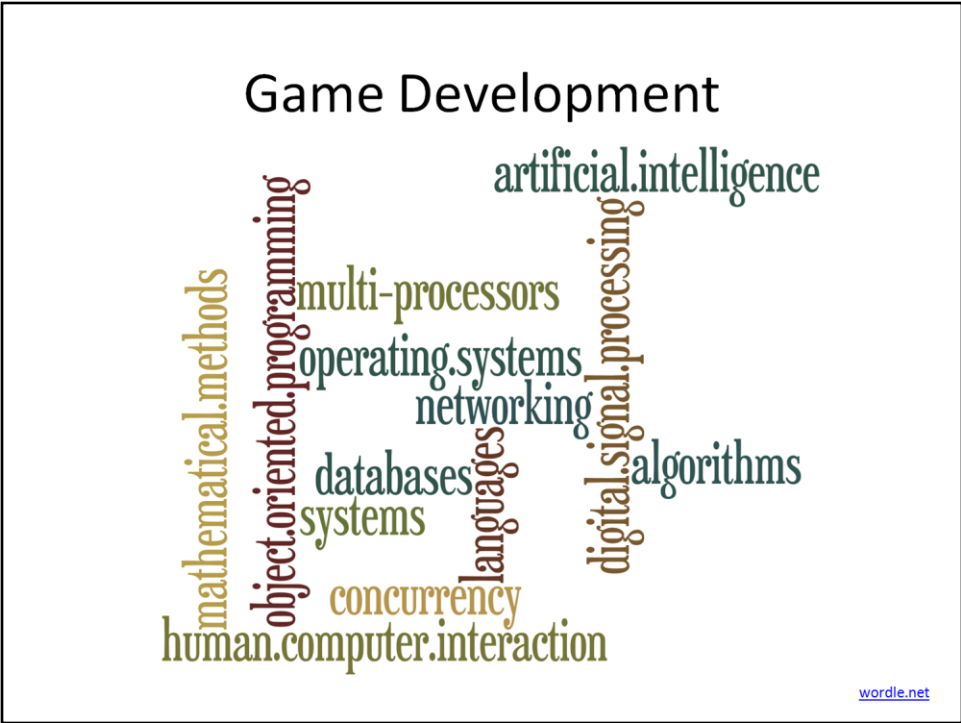
# Industry Trends

- Data in cloud, persistence, levelling-up
- Episodic and user generated content
- Simple natural interfaces 
- Hyper-realism
  - Photorealistic, stylised
  - Can machines simulate interactive reality?



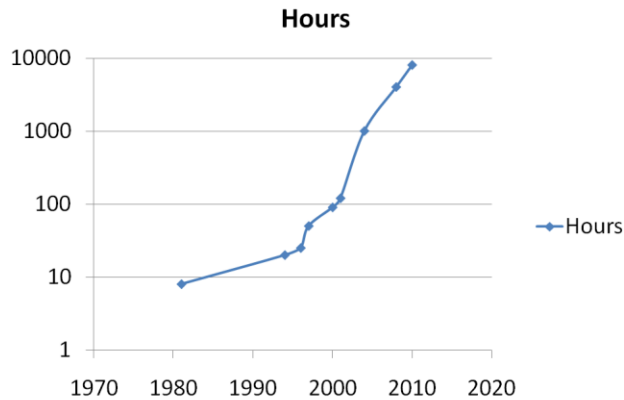
# Real-time

- Define
  - 10hz : Interactive
  - 24hz : Film
  - 30hz : Games
  - 60hz : High refresh rate critical games
  - 120hz : High quality 3D stereoscopic games
  - Needs to be constant, no spikes to break immersion
- Input
- Latency



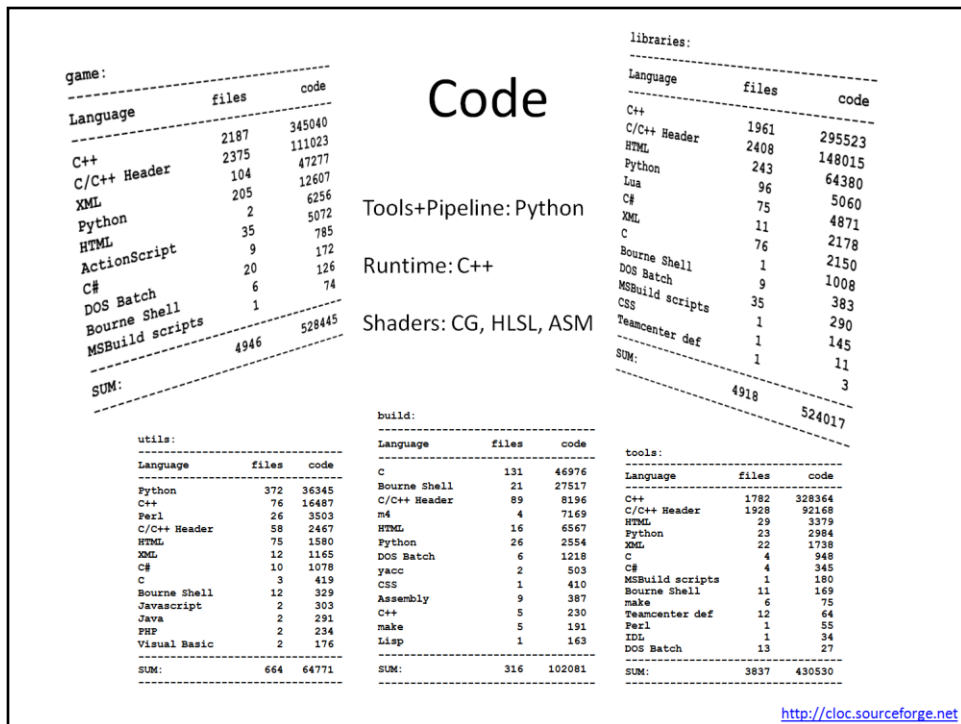
Application of many computing topics

# Programmer Development Time



- Exponential increase in programming time per game

This graph shows the increase in number of man-hours spent programming on various games over the years. The logarithmic hours scale shows just how dramatic this has been with early games being completed with a day's worth of programming, increasing to over 900 hours in the last 12 years. Hikes in the graph can be seen to follow generations of game console hardware where leaps in power and flexibility demanded greatly increased development effort. For example, the introduction of 3D graphics capable consoles in the mid/late nineties, and the high definition, multi-processor era of current consoles.



From a snapshot in mid 2009, internally developed code in the studio amounts to approximately 1.5 million lines of code plus 4.5 million lines of externally contributed source, such as middleware. These break down into roughly a million lines of run time game and library code, supported by ½ million lines of offline utility, build and tools code. cloc doesn't capture all stats though, missing build scripts, shader files, etc.



## Team Size

- Doom (1993)
  - ~10 developers
- Harry Potter (2005)
  - ~70 developers
  - Brook's vs "Potter's" law
- Split/Second (2010)
  - Peak ~150 dev staff
  - Outsourcing
  - Short term contracts



Courtesy Michael Carr

Team size has also increased dramatically to account for increased development time doubling on average every 5 years. Fred Brook's law states that adding coders to a project that is late will make it later. In the case of Prisoner of Azkaban, Potter's law prevailed, where project scope was cut to make the film's release date and share marketing budget.

# Methods

- AGILE
  - Small focused teams, responsive to changes
- Wiki use
  - Efficient communication portal between teams
  - Tasks, blogs, reports
- Automated testing infrastructure
  - Test driven development
  - Unit tests
  - Continuous code & asset builds validating content
  - Render & profiling deviation tests

As development effort has increased dramatically, team size has only increased at a lesser rate. So, more efficient working methods have had to be adopted.

# Stages

## Pre-production

1. Art style, x-movie, R&D, proof of concept
2. Tools production, mature pipeline, vertical slice build

## Production

1. Large art/design/outsourced content team
2. User testing and feedback
3. Optimization, polish, quality assurance testing

## Post-production

- n. Downloadable content (DLC), community, patch

# Platforms

- Focus
  - Relatively aligned
  - Outsourced ports



Platform	Xbox 360	Playstation 3
CPU	3.2 GHz 3 CPU 2 threads each	3.2 GHz 1 Core (2 threads) + 6 SPUs
GPU	shader model 3	shader model 3
Main RAM	512 MB	256 MB
Bandwidth	21.2 GB/s	25 GB/s
VRAM	10 MB (E)	256 MB
Bandwidth	256 GB/s	25 GB/s

For this generation of consoles we've chosen to develop on relatively aligned console platforms with similar levels of performance and features.

This allows us to focus on generating the best technology for the game without worrying about the details of ports to other platforms, which are gratefully outsourced.

360 is essentially a symmetric parallel processor comprising 3 CPUs with 2 threads each and a unified memory model. Whereas PS3 is an asymmetric parallel processor with one primary processing unit and 6 useful cell synergistic processing units. Video memory is much faster on 360, at the cost of being much smaller.

# Bottlenecks

- Build
  - Code, Assets, Live Update
- Memory
  - Media, RAM, VRAM
- Simulation
  - AI, animation, physics
- Rendering
  - Geometry
  - Shading

## Not Bottlenecks (unless...)

- Game logic, control, progression
  - Increasing to 1000s of entities
- Simple joypad input
  - Image processing, skeletal tracking, biometrics
- Audio
  - Speech recognition, synthesis, voice location

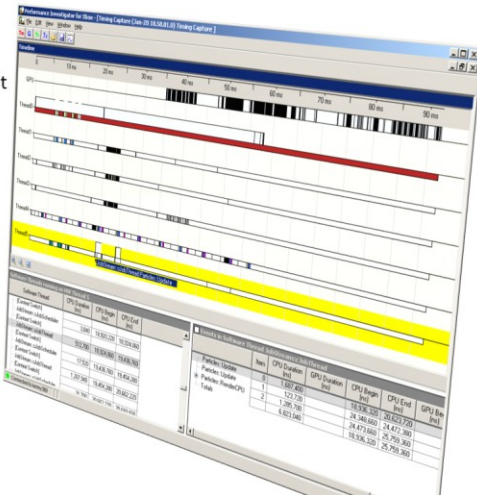
# Optimization

- Knuth/Hoare
  - ‘Premature optimization is the root of all evil’
- ‘Moore’s law is dead’, [Gordon Moore](#)
  - Increase use of parallelism to multi-core and many-core
- Amdahl’s law
  - Parallel speed up is limited by sequential portion of process
- Gustafson’s law
  - Sequential portion relatively small when massively parallel

Gustafson’s law is most relevant to video game performance challenges such as rendering.

# Game Parallel Processing

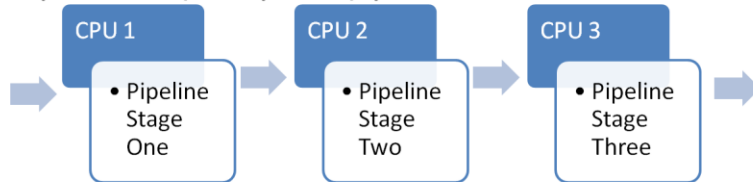
- Directed graph of task & data parallel steps
- Data Parallel
  - Performs same task on
    - Separate blocks of a large dataset
  - Physics, Geometry & Shading
- Task Parallel
  - Performs different tasks on
    - Same or separate data
  - Audio, Particles, Visibility
- Instruction Parallel
  - Pipelining, superscalar
  - Out-of-order execution (n/a)



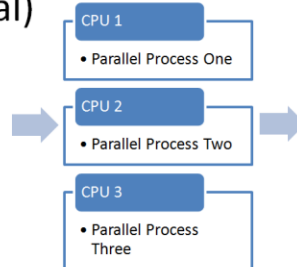


# Pipelining

- Pipelined (temporal) parallelism

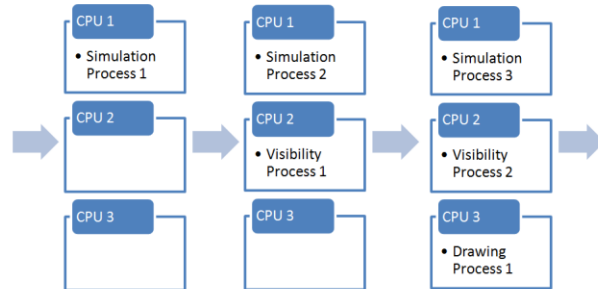


- Parallel (spatial)



Ideally we'd like to process all tasks in the game in parallel on separate CPUs, sometimes known as spatial parallelism, but there's typically dependencies between tasks which prevent this. Pipelined (or temporal) parallelism permits sequential processing of dependent tasks in parallel. Above CPUs 1-3 denote separate parallel processing units.

# Pipelined Game Processing



- Increases throughput, but introduces latency

Taking a simple game's simulation and rendering loop for example, we know the drawing process is dependent on visibility processing and that in turn depends on simulation processing. If we pipeline these processes across three stages, we can achieve parallel processing of these tasks. This yields the ideal throughput performance speed up of performing the three tasks in parallel, but the lag that has been introduced means it takes three stages to see the response of simulation updates.

# Latency

- Stages of an interactive loop
  - Player
    - Sensory impulses -> reaction time -> motor control
  - Input
    - Physical input -> device capture -> signal process
  - Simulation
    - Interpret input -> compute response -> update state
  - Render
    - Dispatch drawing commands -> video signal -> display lag
- 1 to 5 frames depending on
  - Required responsiveness
  - Device and hardware characteristics

## Elements of a Blockbuster

- Shading
- Lighting
- Physics
- Particle
- Cameras
- Speed

# Car Shading

- 2 Tone Paint
  - Fresnel
  - Clear coat
  - Dynamic reflection
- Damage
  - Smoke
  - Lacquer scratches
  - Scrapes
  - Glass



# Lighting

- Deferred Shading
  - Reduces per-pixel shading cost to only visible surfaces
- First Pass
  - G-Buffer
- Shading Pass
  - Dynamic
  - Many lights



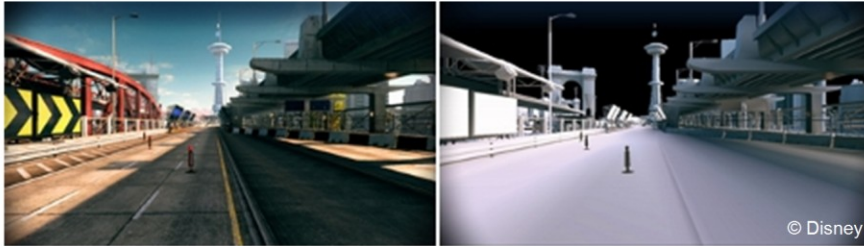
# Lighting

- Rigs
  - Day
  - Night



# Lighting

- Some lighting elements baked offline
  - Static global illumination
  - Too costly to compute (our R&D work)





# Lighting

- Tiled classification of image elements permits reduced shading cost
  - Soft shadow edge filtering
  - Geometry edge anti-aliasing



# Rock Blast

- Grey box
- Prototype simulation
- Billboard particles



# Rock Blast

- Textured
- Validate fracture visuals



# Rock Blast

- Apply particle effects



# Rock Blast

- Combine with environment



# Rock Blast

- In game
  - Debris away from track, interactive boulders, particles



# Particles

- Non-interactive particles add to the visual composition



# Interactive Physics

- Distorting particles
  - Apply turbulence force of volume displaced by cars





# Rubber Neck Camera

- Accentuate highlights with bullet time camera zoom



## 3 Way Track Changes

- Destructive events change track layout
  - Updating AI car paths and collision geometry



© Disney

# 1.5Km of Destruction

- 1000+ animating joints processed in parallel on SPUs



# 1.5Km of Destruction

- Pushing the edge of floating-point precision for collision volumes



# Velocity



# Questions?

- Thanks to the Black Rock Studio Team

- [Kenny.Mitchell@disney.com](mailto:Kenny.Mitchell@disney.com)
- [Dawn.Beasley@disney.com](mailto:Dawn.Beasley@disney.com)

