License an Engine
Or Create Your Own?

Dietmar Hauser
Head of Console Technology
Sproing Interactive Media GmbH

Some Background…

- Located in Vienna, Austria
- ~90 employees
  - Multiple parallel projects
- Independent since 2001
- All platforms, all genres
- Proprietary engine…
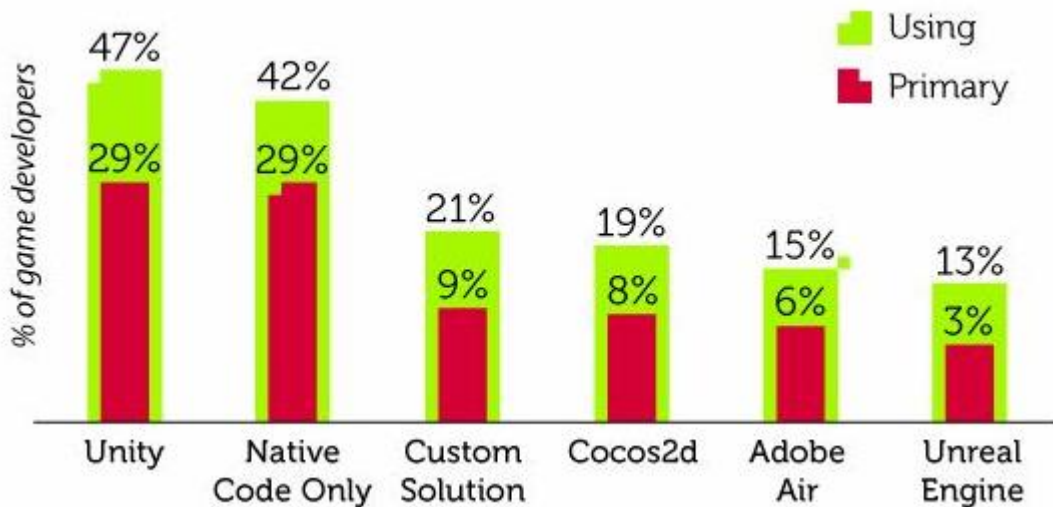   …not always used

Athena 2

- 100% native, C++
- Modular and portable
- Augmented by middleware
- Improved gradually
- Slim dedicated budget

What are the others doing?

No right way: almost 50:50 split

Source: Developer Economics | State of the Developer Nation Q3 2014

Part 1: Strategy

- Investment vs. cost
  - License fee vs. development effort
  - Hiring vs. training experts
  - Flexibility vs. stability

- Switching engine kills investments!
  - Especially during a project

Engines are a safe bet, right?

RenderWare

- Very successful in the early 2000's
  - Esp. on PS2: GTA3, Burnout, …
- Owned by Canon, bought by EA in 2004
  - Sales stopped, support & updates decayed

What about Unreal?

- Highly respected developer
- 2012: Tencent buys 48.4%
  - Unreal business model changes
  - Some key people leave
- Tencent is very big
  - Expands aggressively
  - Knows of all Unreal powered games

What about Unity3D?

- Dominates mobile 3D games
- Excellent marketing efforts
- Entirely investment funded
  - Undisclosed amount in 2014
  - Constant buy-out rumors
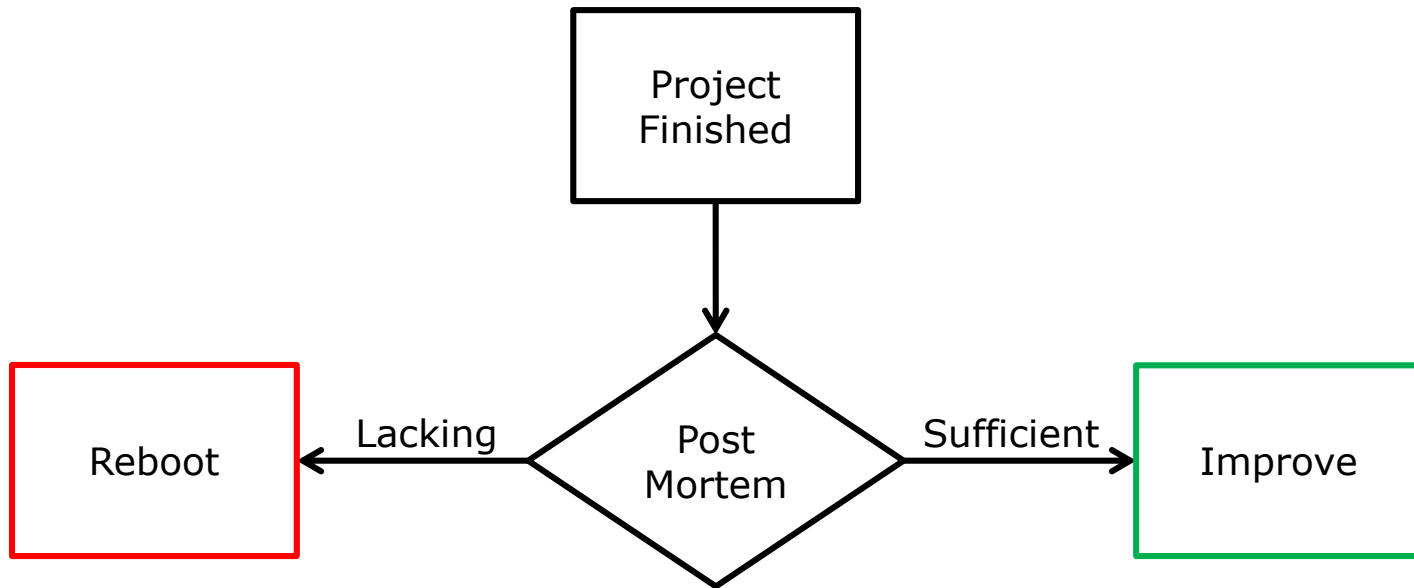- Cloud services tighten dependency
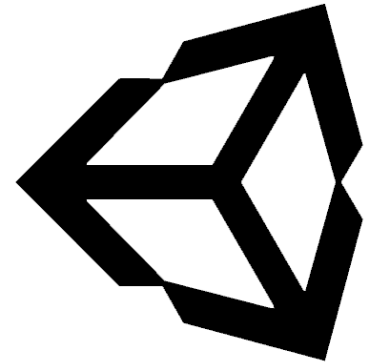
Sproing's middleware criteria

- Shopping list
  - Fixed or predictable costs
  - Source code access
  - Premium support option
  - Hookable syscalls
    - Esp. dynamic memory

Part 2: Maintenance

Revolution

- A complete Athena rewrite
  - To get a clean slate for teh future™
  - Rewrite is isolated, ambitious project
  - Added new platforms, dropped old ones
  - Focus on breadth and not depth
  - Great loss of practical functionality
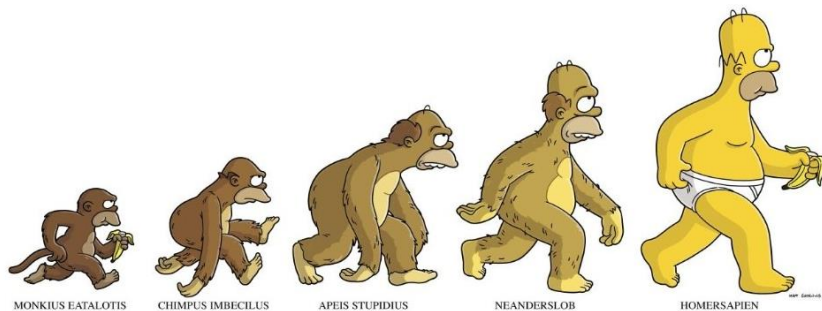 - Half finished features continue to haunt us

Evolution

- Lessons learned
  - Develop framework + game in sync
  - Focus on incremental updates
  - Isolate critical changes with branches
    - Allow parallel systems
  - Refactoring is extremely dangerous!
    - Very appealing
   - Often turns into "refucktoring"

Should have used an engine!

- Different approach, same problems
    - Decisions are outsourced
    - Support for old versions decays
    - No good upgrade path to new versions
        - Customizations and knowledge are lost
        - Functionality is often also lost
        - Subtle changes cause maintenance effort
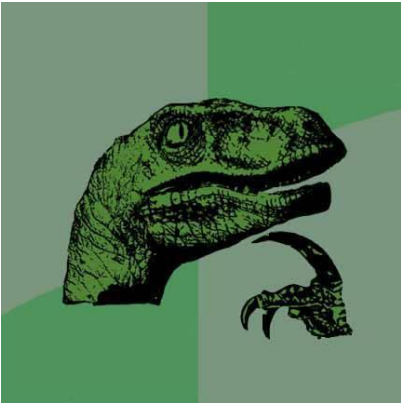   - Tough to add new platforms to old projects

Proficiency growth (idealized)
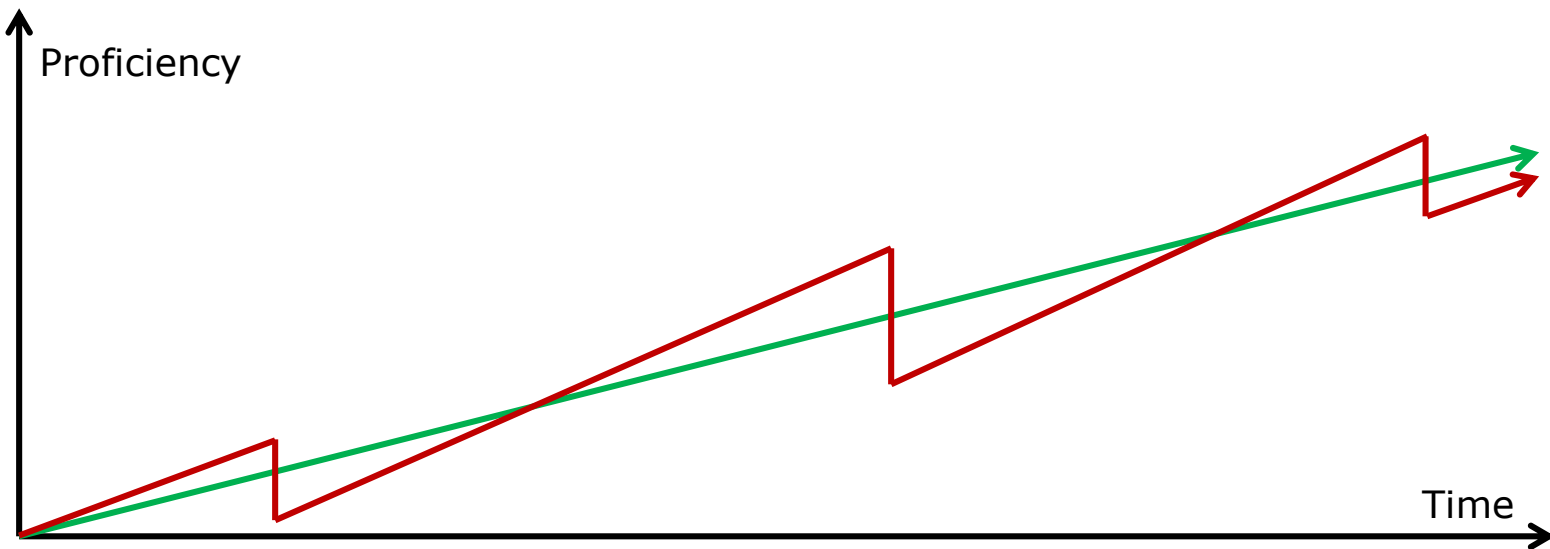
What does this mean?

- Software engineering is hard!
- The bigger the change, the harder the fall
- Outsourced problems are still problems

Choose wisely:
- Revolution: Fast, but dangerous
- Evolution: Slow, but controlled

Part 3: Tools

- Make making games possible
- The target of your investment
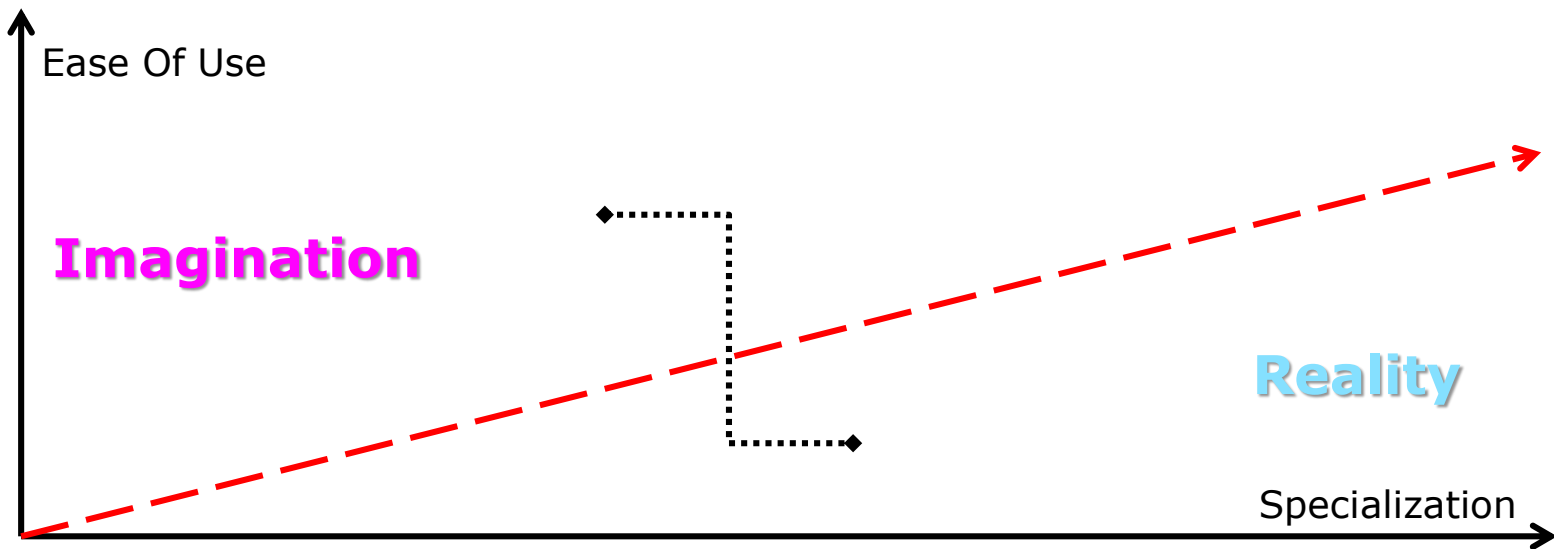- Come in all shapes and forms
- Expensive
- Vital
- But…

Every Tool Sucks!

- They are too hard to use
- Don't do what's expected
- Cause frustration
- Decrease productivity
- …until they don't anymore

"Possible Tools Space"

Ease Of Use

Imagination

Reality

Specialization

What does this mean?

- "Marketing" is key for tools acceptance
  - Needs to reflect reality
  - Needs to engage users emotionally
  - Both help over the initial hurdle
- Tools vendors do this, often unintentional
  - Unity3D does it best
  - In house development (usually) worst

Conclusions

- Find your strategy
- Pick tools that fit it
- Stick with it
- Gradually improve
- Manage expectations
- If all fails, reboot

KEEP
CALM
AND
DRAW
CONCLUSIONS

Contact Information

dietmar.hauser@sproing.com

@Rattenhirn

https://www.sproing.com

https://fb.me/sproing

@SproingGames