MIKE ACTON @ INSOMNIAC GAMES

WHAT I WISH NEW ENGINE PROGRAMMERS KNEW

HUGE DEMAND FOR GOOD ENGINE PROGRAMMERS, BUT...

NEW ASSOCIATE ENGINE = -2YRS

LOTS OF BAD ADVICE...

Concepts Learning to Program Programming Languages Computer Programming

What are the five most important programming concepts?

- **Single responsibility:** Every module, class, function, method should have a single responsibility. If you have a method called “convert_and_persist” you have done something wrong. it should be one function to convert the date and one handling persistance.

- **Single responsibility:** Every module, class, function, method should have a single responsibility. If you have a method called “convert_and_persist” you have done something wrong. it should be one function to convert the date and one handling persistance.

NONSENSE. SEE: MADD

- **Don’t reinvent the wheel:** If there are libraries used by thousands of other developers, then use it as well and don’t write the stuff yourself. Open source liraries are in 99% of the cases more efficient than what you would write because a huge bunch of people smarter than you optimized that code.

- **Don’t reinvent the wheel:** If there are libraries used by thousands of other developers, then use it as well and don’t write the stuff yourself. Open source liraries are in 99% of the cases more efficient than what you would write because a huge bunch of people smarter than you optimized that code.

NONSENSE. REAL WORLD IS NOT ONE SIZE FITS ALL.

5. Back when you were learning to read and write, you spent far more time reading than writing. Current pedagogy emphasizes writing code over reading code --- what was the last programming class you took where you started with an existing 1k line program and *read* it? As a professional, you'll be spending most of your time reading code you didn't write in order to find a bug or add a feature. It's a skill worth developing early.

5. Back when you were learning to read and write, you spent far more time reading than writing. Current pedagogy emphasizes writing code over reading code --- what was the last programming class you took where you started with an existing 1k line program and *read* it? As a professional, you'll be spending most of your time reading code you didn't write in order to find a bug or add a feature. It's a skill worth developing early.

OK BUT MISLEADING. SKILL IS INFERRING PURPOSE AND CONSTRAINTS.

1. You don't have to recode anything if you plan and design it first

1. You don't have to recode anything if you plan and design it first

LOL

3. Write in such a way so that the next programmer picking up your project can actually read and understand your code

3. Write in such a way so that the next programmer picking up your project can actually read and understand your code

OK BUT MISLEADING. DOESN'T MEAN ASSUME NEXT PROGRAMMER IS AN IDIOT.

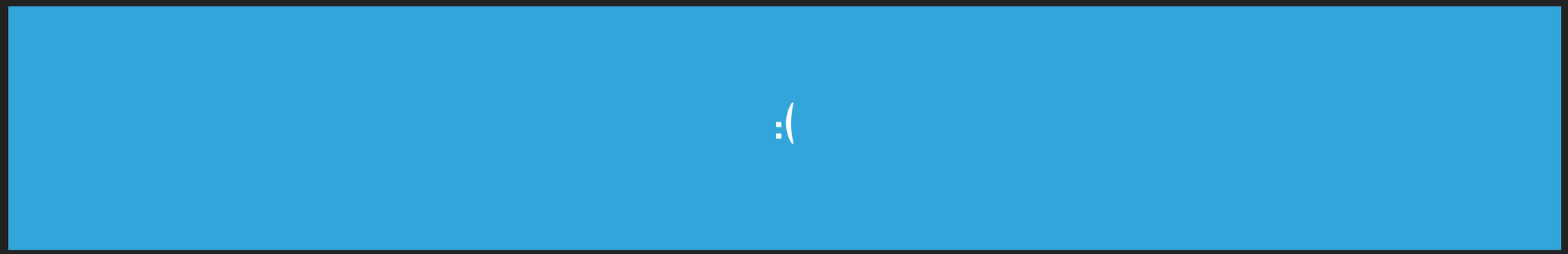
1. Abstraction. Realise that programming is about computation and computation has nothing to do with electronic computers. Electronic computers are just the fastest current way to perform computations. That is what abstraction is about. Abstraction means we can be machine independent in our computations.

1. Abstraction. Realise that programming is about computation and computation has nothing to do with electronic computers. Electronic computers are just the fastest current way to perform computations. That is what abstraction is about. Abstraction means we can be machine independent in our computations.

FFS. YOU ARE ENGINEERING FOR REALITY NOT AN IMAGINARY FAIRY TALE LAND

2. Single level of Memory. Memory is single level and programmers should not be concerned with how memory is allocated or released or moving data between different levels of memory. The ideal Turing machine only had one level of infinite memory. Memory levels have nothing to do with the computational model (as above, most things we do with electronics is just implementation detail). Virtual memory realised this early and it is invisible to programmers. That means programmers should not be concerned with IO to move between memory layers - that is IO at a physical level, but by abstraction again, be concerned with problem-domain IO that gets input from a user and gives them feedback. Processor registers should also never be visible at any programming level, including machine-oriented languages such as assembler. The Burroughs B5000 computer realised this and did not have programmer-visible registers. It was also the first commercial machine with virtual memory. So a machine that blurs memory levels has been around for over 50 years. (B5000 is documented on Wikipedia, but now you can play with it for yourself MCP Express Is Here!)

2. Single level of Memory. Memory is single level and programmers should not be concerned with how memory is allocated or released or moving data between different levels of memory. The ideal Turing machine only had one level of infinite memory. Memory levels have nothing to do with the computational model (as above, most things we do with electronics is just implementation detail). Virtual memory realised this early and it is invisible to programmers. That means programmers should not be concerned with IO to move between memory layers - that is IO at a physical level, but by abstraction again, be concerned with problem-domain IO that gets input from a user and gives them feedback. Processor registers should also never be visible at any programming level, including machine-oriented languages such as assembler. The Burroughs B5000 computer realised this and did not have programmer-visible registers. It was also the first commercial machine with virtual memory. So a machine that blurs memory levels has been around for over 50 years. (B5000 is documented on Wikipedia, but now you can play with it for yourself MCP Express Is Here!)



- The compiler is your friend
- Premature optimization is the root of all evil
- Big-O notation

- The compiler is your friend
- Premature optimization is the root of all evil
- Big-O notation

MISLEADING...

- The compiler is your friend
- Premature optimization is the root of all evil
- Big-O notation

THE COMPILER IS A TOOL. IT'S NOT MAGIC.

- The compiler is your friend
- Premature optimization is the root of all evil
- Big-O notation

DON'T CONFUSE GETTING THE CLOWNS OUT OF THE CAR WITH OPTIMIZATION

- The compiler is your friend
- Premature optimization is the root of all evil
- Big-O notation

BIG-O: C IS VARIABLE AND VERY LARGE. NOTHING WRT CONCURRENCY.

WHAT COULD I BE EXPECTING...?

TECHNICAL SKILLS

- How does the OS work?
- How does the CPU work?
- How does the GPU work?
- How do the memory busses work?
- SIMD
- Read/write assembly
- Research area expertise
- Etc.

BUT THE REAL PROBLEM IS MUCH DEEPER...

ESSENTIALLY
INCOMPETENT IN THREE
FUNDAMENTAL AREAS...

BASIC COMPETENCIES

- Practice
- Reasonable defaults
- Problem solving

QUICK CRASH COURSE

(IT'S NOT HARD)

PRACTICE

WHAT IS PRACTICE?

- Explore gaps in knowledge
- Ephemeral
- Not competitive
- Not research
- Not a project
- Daily.
- You can fit in 30 minutes. Plan!

REASONABLE DEFAULTS

SHORTCUTS TO FIRST PASS

- Linear search through array
- FIFO managed by incrementing integer
- Store by type
- Multiple by default
- Explicit latency and throughput constraints
- Version serialized data
- Allocators: Block, Stack, Scratch
- Model target manually first (cheat)
- Index look aside table

PROBLEM SOLVING

WEAK PROBLEM SOLVING EASILY MOST SHOCKING AND BIGGEST ISSUE HOLDING PEOPLE BACK

SOLVING THE -RIGHT- PROBLEM

- An iterative and inter-related process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

SOLVING THE -RIGHT- PROBLEM

- An iterative process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

ALWAYS CHANGING.
WRITE THEM DOWN.

SOLVING THE -RIGHT- PROBLEM

- An iterative process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

CONTEXT IS THOSE THINGS THAT CAN BE ASSUMED KNOWN

THE MORE CONTEXT YOU HAVE THE BETTER YOU CAN MAKE THE SOLUTION

E.G. LADDERS

DIFFERENT PROBLEMS REQUIRE DIFFERENT SOLUTIONS

IDENTIFY USERS' NEEDS

- Watch them work
- What data are they transforming?
- Describe concrete goals in plain language (not features)

IDENTIFY AND BALANCE CONSTRAINTS

- Iteration time
- Learning
- Size
- Speed
- Correctness

3 COMMON CONTEXT TRAPS

THE WHAT-IF GAME

- Do you have an actual concrete example?
- Can you test it?
- How much experience do you have with the problem?

- Solving problems you don't have creates problems you definitely do.
- Future proofing is a fool's errand
- Future is unknowable
- You will be better positioned in the future anyway
- People are creative

OVER SIMPLE

- You can't make a problem simpler than it actually is
- Other people "ruining" your design with hacks? The problem is not the hacks.
- Do you know where the data is coming from?
- ...how long it takes to create?
- ...how complex the process to create it is?
- Broken models that don't fit. The hard parts are the job.

OVER COMPLICATED

- Too generic
- Trying to solve by storytelling
- Generic as an excuse not to actually solve the problem
- Generic as an excuse not to talk to users
- Degenerates to yet another compiler

SOLVING THE -RIGHT- PROBLEM

- An iterative process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

VALUE

- Identify specific, concrete, measurable value
- Make the business case
- It's not about being a suit. It's about not being irrational with your most valuable resource (your time)
- There are more things than you can possibly do

BUSINESS CASE

- Sales is a big part of teamwork. Show value for...
- Cost of entry
- Development time
- Increased satisfaction (internal or external)
- Learning
- Enabling exploration
- Size
- Speed

BUSINESS CASE

- Articulate real value targets...
- How much iteration time?
- How much speed?
- Reason about value...
- How much is 1 man-month of production time worth?
- How much is 1ms from the frame worth?

SOLVING THE -RIGHT- PROBLEM

- An iterative process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

COST

- Predicted development cost (generally wrong)
- Opportunity cost (better vs. new)
- Maintenance cost (the REAL cost)

IF YOU CAN'T REASON ABOUT THE MAINTENANCE COST, YOU'RE SPENDING RESOURCES YOU DON'T HAVE.

SOURCES OF MAINTENANCE COST

- Maintaining understanding of the data
- Changing requirements
- Communicating constraints
- Untested transforms
- Unexpected use cases
- Dependency changes
- Bad inputs
- Usage training
- Infrastructure
- Any changes whatsoever

BUILD VS. BUY

HOW WELL CAN YOU REASON ABOUT VALUE/COST?

SOLVING THE -RIGHT- PROBLEM

- An iterative process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

REALITY IS NOT A HACK YOU NEED TO DEAL WITH TO SOLVE YOUR ABSTRACT, THEORETICAL PROBLEM. REALITY IS THE ACTUAL PROBLEM.

PLATFORM

- No such thing as "platform independent"
- RTFM
- x64 vs. ARM
- NVIDIA vs. ATI vs. PowerVR
- OpenGL vs. OpenGL ES vs. DirectX 11 vs. Vulkan vs. ...
- Windows vs. Linux
- Cannot abstract h/w. Can organize commonalities.

PLATFORM

- Shared device access costs...
- Caches
- RAM
- HDD, BDD
- Network

PLATFORM – TOOLS

- Know your tools
- E.g. compilers
- How does it work?
- What does it do?
- What does it output exactly?
- How can you influence the output?
- You are responsible for the output.
- Tools that get in the way of understanding the details are bad tools.

SOLVING THE -RIGHT- PROBLEM

- An iterative process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

EVERYTHING IS A DATA PROBLEM

IF YOU DON'T UNDERSTAND THE DATA YOU DON'T UNDERSTAND THE PROBLEM

THE PURPOSE OF ALL PROGRAMS AND ALL PARTS OF ALL PROGRAMS IS TO TRANSFORM DATA FROM ONE FORM TO ANOTHER

GATHER DATA ON DATA

- Sample input and output to problem
- What does it look like? (Visualize)
- What is read and written over time? Access patterns?
- When is data accessed statistically relative to other data?
- What values are common? Outliers?
- What ranges are common? Outliers?
- What data causes branches? Statistics on states

GATHER DATA ON DATA

- Sample input and output to problem
- In vivo tools
- In vitro tools
- Common tools for this analysis are poor or non existent

DATA – GET THE CLOWNS OUT OF THE CAR

- Recursively remove unnecessary work from…
- Inner loops
- Per frame
- Over time
- Per zone/level/etc.
- Per game instance
- Offline builders…
- Tools…

REVIEW QUICK CRASH COURSE EXPECTATIONS...

SOLVING THE -RIGHT- PROBLEM

- An iterative and inter-related process of understanding...
- CONTEXT
- VALUE
- COST
- PLATFORM
- DATA

BASIC COMPETENCIES

- Practice
- Reasonable defaults
- Problem solving

NEW ASSOCIATE ENGINE = 6MO