

# [Game] Programming I Didn't Learn In School

presented by

Anton Gerdelan

Trinity College Dublin

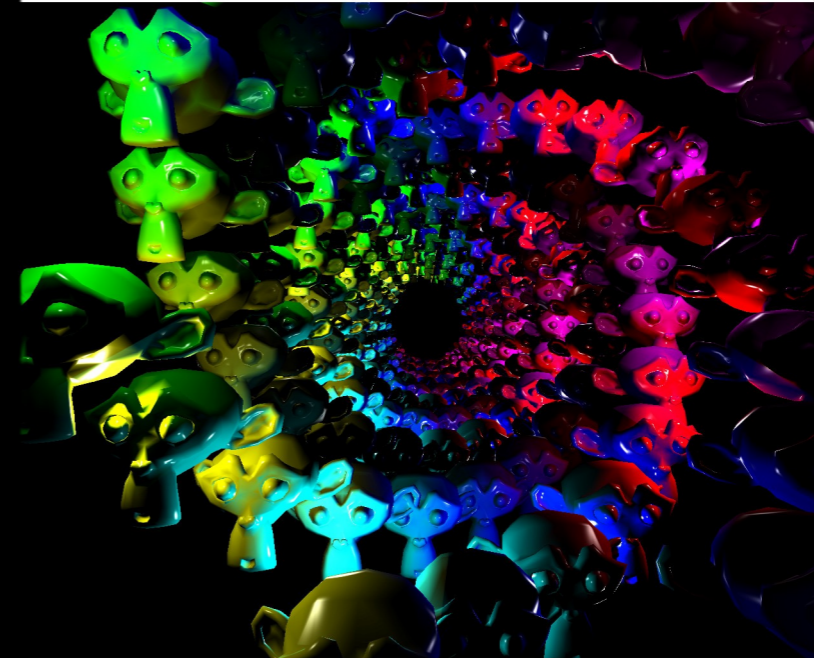
[<gerdela@scss.tcd.ie>](mailto:gerdela@scss.tcd.ie)

[antongerdelan.net](http://antongerdelan.net)

me

- computer graphics research, Trinity College Dublin, Ireland
- lectured graphics course at Trinity College
- learned shader programming whilst working with Stefan Petersson here
- lots of coding support for graphics courses

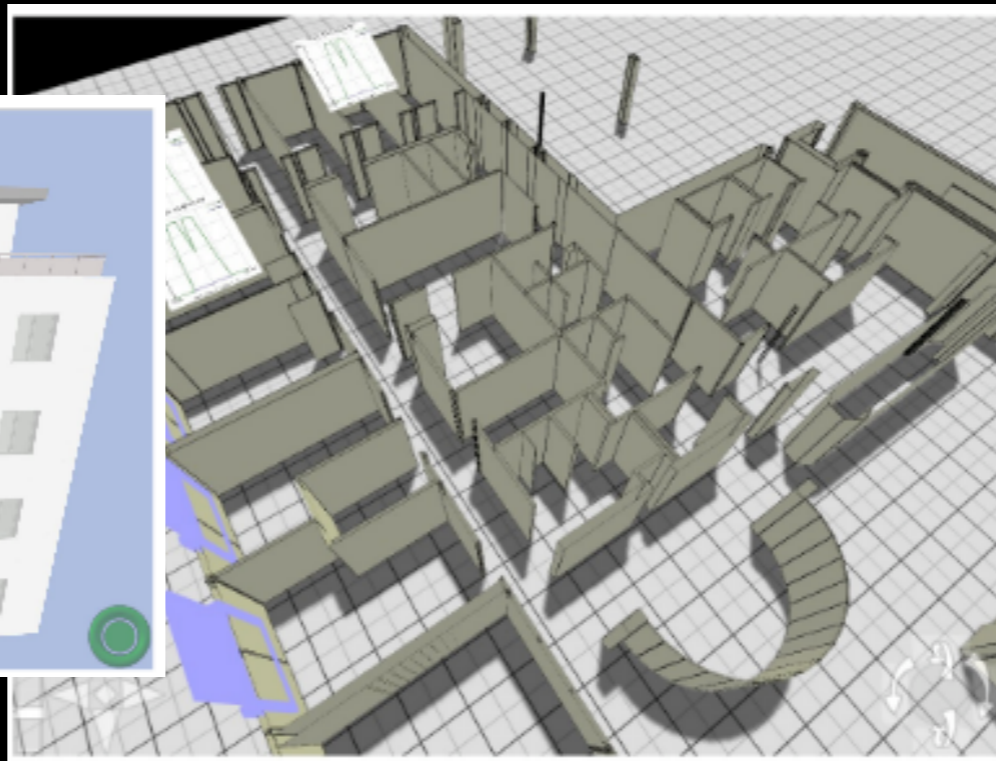
# **Anton's OpenGL 4 Tutorials**



**Anton Gerdelan**

# research projects

- AI behaviour. Steering. Vehicles. Fuzzy/GA
- WebGL architectural models
- attempting: realistic human rendering / Unreal engine
- motion capture. animation models.
- perceptual experiments





# The Life Cycle of Guinea Worm Disease

# Life Cycle of the Programmer

## 1 The cycle starts...

Seeking relief from pain, sufferers soak a blister with exposed worm in nearby water source. On contact with water, the worm bursts, releasing hundreds of thousands of immature first-stage larvae into the water.

3 Another person drinks the water containing the water fleas with the infective larvae. The water fleas are digested, releasing the larvae into the stomach.

## 7 The cycle continues...

4 The larvae, which resist digestion, migrate to the small intestine and penetrate the intestinal wall into the body cavity, where they grow into worms and mate.

- Start with some basic programming; “transform X into Y”
- Spend 10 years learning ALL the formalised design methods
  - taking on complexity is almost never discouraged
- “90% of my time is spent refactoring and structuring”
- yearn for early days
- At some point forced to make something the simple way again
  - ...and it's fun...and quick to make...
- The unusual commonality of the “moment of realisation”

2 Tiny water fleas ingest the larvae which, molt twice, become infective stage larvae. The process takes about 2 weeks.

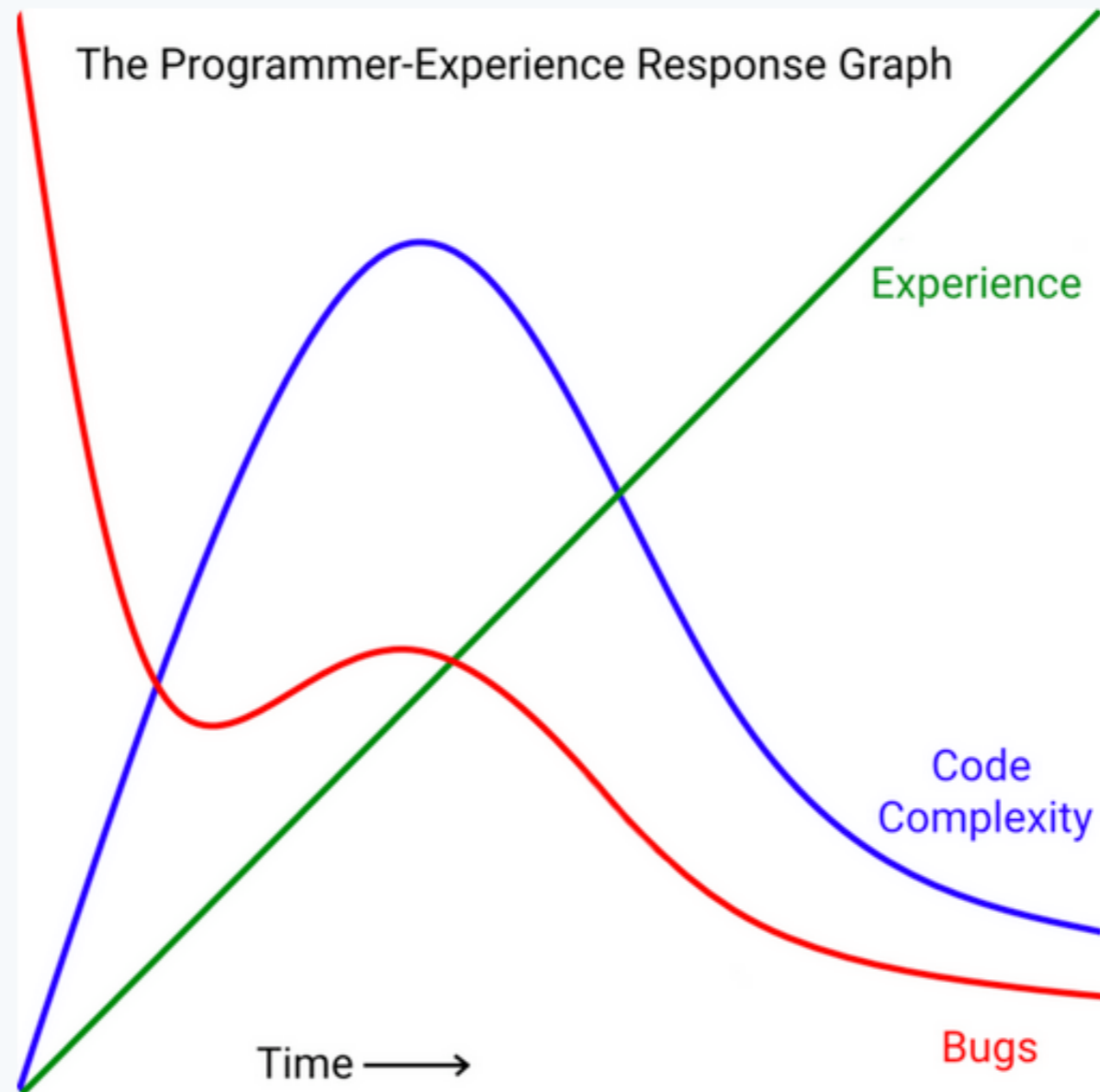
5 Fertilized female worms, up to 3 feet long, move through connective tissue to various areas of the body, usually the lower limbs.

6 Approximately a year later, after the larvae were ingested, the worm forms a painful blister near the skin surface.



Keith O'Connor @keithoconor · Oct 2


## My completely scientific Programmer-Experience Response Graph



(tech. lead Ubisoft Montreal)

has a great blog with lots of advice for aspiring game programmers [fragmentbuffer.com](http://fragmentbuffer.com)





“There was a time, boy, when I searched for  
steel, when steel meant more to me than gold or  
jewels”

*–Thulsa Doom, Conan the Barbarian (1982)*

# Why is complex bad?



- Takes too long to write (and read)
- Refactoring and the fallacy of the ducks
- Does not respect the computer
- Unhelpful abstraction
- We get very carried away with notions

# Quandary

- We change our style ideas every year or two
- I *could* present my “this year” ideas
- Another opinionated rant from an academic about programming games?



# Interview Prompts

“What would you tell yourself about programming **if you could go back in time?**”

“What advice can you give about **keeping code simple** or working expediently?”

“What are **typical mistakes** that new graduates make, that they would benefit from hearing now?”

“Is there one **reference/book/person** that you recommend students read or follow?”

Mini Interview 1

# Niklas Lundberg

Game engine programmer  
Avalanche Studios  
Stockholm

“Don’t bother with object oriented programming, it's not helpful”



“Try to keep functions having 1 task,  
not multiple  
(depending on extra passed in  
booleans etc.)”

“Learn to program a real  
machine, not a virtual  
machine.  
e.g. use C, not java”

“Don’t keep global state.  
Functions should have all  
inputs they need”



- “watch all videos in this series:

<https://handmadehero.org>

- and don't proceed to the next until you understand each episode 100%
- requires basic C knowledge
- has a mini course for that too, but you will need some more”

# Mini Interview 2

# John Romero



“Keep your code **absolutely simple**. Keep looking at your functions and figure out how you can **simplify further**.”



“Write your code **for this game only** – not for a future game. You’re going to be writing new code later because you’ll be smarter.”

“It will take you **10 years of constant programming** and pushing yourself before you will be able to do something important. Study coders you respect and see how long they were programming before their big hit. Read **Outliers by Malcolm Gladwell**. There are no shortcuts.”

“Try to **code transparently**. Tell your lead and peers exactly how you are going to solve your current task and get feedback and advice. Do not treat game programming like each coder is a black box. **The project could go off the rails** and cause delays.”

“Programming is a creative art form based in logic. Every programmer is different and will code differently. **It’s the output that matters.**”

“If you’re making a game with a small team, don’t use GIT – use SVN. Try not to branch. Always keep your code current as often as possible.

**Everyone should be able to make a full build and run the game at any time.”**



“Programmers should **code as if the QA team does not exist**. When you find a bug, **fix it immediately**. Do not code further. You risk your codebase **depending on that bug**. id Software did not have a QA team before I left after Quake 1.”



## Mini Interview 3

# Ivan-Asseň Ivanov

CTO

Haemimont Games  
Sofia, Bulgaria



Nasz statek handlowy wyeksportował tropikańskie towary warte 7 314\$.

Wrz, 1939

\$22 541 550 93%

Wojny światowe



Rancza 1/4

Wyeksportuj: Cygara 1260/5000



“There's a saying that making a game is like **doing major reconstruction work on your airplane while you're flying it**; I heavily recommend having an old airplane needing reconstruction in the first place!”

“I see some sense in the classical *one class per real-world concept* OOP design style in the highest levels of gameplay code, **e.g. having one object of class "Unit" for each, uhm, unit in the game,** and where performance is the least concern.”

“I see some value in the idea of "design patterns" as a common language; we say *let's use the reasons pattern* and everyone knows what we're talking about here.

I don't get the **religious attitude towards the book**, and the treating of its list of patterns as end-all, be-all - something I'm sure **the authors never intended.**”



“I never got the point of UML.  
**Pick a high-enough level  
language and you won't need  
diagrams** - you'll fit the "big  
picture" on a screen, then zoom  
as appropriate.”

“A modern game contains many different subsystems, and what is true of code that is executed **once per (one of ten thousand) object per frame** is not as applicable of code that is executed **once in a few seconds per human player**.

Writing code **carefully considering the hardware** is very rewarding, and **very much necessary in the first case**; unfortunately, it's also much harder, and working via **abstractions can be forgiven in the second.**”

**“The first thing new recruits need to unlearn is the love for their own code.** New programmers love nothing more than to produce code - pages and pages of elaborate, complex code. But **code is a liability, not an asset.** The job of a programmer is to think first, and to solve problems of his customers (e.g. designers and artists in a game team) second. Not to produce code. Programmers need to learn to love deleting code more than they love writing new code. Programmers need to learn to let their code go and **mercilessly delete and simplify** when a better, or simpler solution presents itself.”

“I think an important habit to have is to resist the temptation to abstract and create “general solutions” the first time you encounter a problem. It is better to implement simple, concrete solutions to the first and even the second occurrence of a similar problem. **Only the third time you know enough about the problem to think of generalising** - and even then you can't be sure you've seen it all.”

“Anyone with a desire to delve in engine code should get a good understanding of C, a decent understanding of C++, and a cursory understanding of the assembly language of the machine they're targeting.”



“Nowadays it's hard to evade JavaScript - it's not a very good language, but it's ubiquitous in large portions of the IT industry. C# is also a good bet, being embedded in Unity and also potentially useful in the real world, if you decide games aren't for you after all.”

“I like the **Handmade Hero project by Casey Muratori**. It's a series of videos showing how a non-trivial game is made from first principles, without using `_any_` middleware or "game engines". It's very instructional, and Casey is a **gifted comedian just wasting his talent as a programming superstar.**”



“Yes! You know what it is, don't you boy? Shall I tell you? It's the least I can do. Steel isn't strong, boy, flesh is stronger!”

*-Thulsa Doom*