Interviewing. What's that?

Unity Training Academy 2018-2019, #1
Aras Pranckevičius

Generative Art – Made with Unity

Caveat Emptor

- May or might not apply in other places
- Especially in different industries
- I might be talking complete nonsense too!

Yo dude, how did you get a job anyway?

- 2000, Java: friends I was doing gamedev with
- 2001, computer vision / gamedev: journalist who knew I was doing gamedev (from school)
- 2004, C#/databases/…: applied myself
- 2005, gamedev: friends I was doing gamedev with
- 2005, (failed) engine dev: website & techdemos
- 2006, Unity: website & techdemos
    - *Got kinda stuck here :)*

Typical process

Typical process

- Send in your CV/resume & letter
- At-home programming test
- Voice/video interview
- On-site interview
- …
- Profit!

unity

Typical process lately has been changing

- Some interview practices found not useful
  - e.g. whiteboard coding
- More attention to bias, discrimination, …

CV, Resume, Letter

CV/Resume

- Short and to the point
  - Don’t make it longer than 2 pages
- Europass CV → Recycle Bin
- Adapt to the company/position!

Most important things on CV

- Experience & projects
  - github, blog, technical social media, website, portfolio
- This mostly means “spare time / hobby”
  - Bias against people who don’t/can’t do this!
  - Some companies aware of this, some not

(not) the most important thing on CV

- University Experience
    - Sorry, not worth much
    - ...unless it’s kick-arse Uni (don’t exist here in LT)
    - ...or applying for government / research job

unity

Most important things on CV

- Actual hours spent learning stuff
  - “10000 hours” rule has some truth
  - Have to spend time to learn something
  - Given “just Uni assignments” and “I code stuff myself”, most companies would pick the latter
    - Unless they actively take resulting bias into account. Many don’t.

Portfolio: code

- Just make a github account, it’s free
- Put your code there
- Do it now
- Just dowit

unity

”I don't want to put my code online because...”

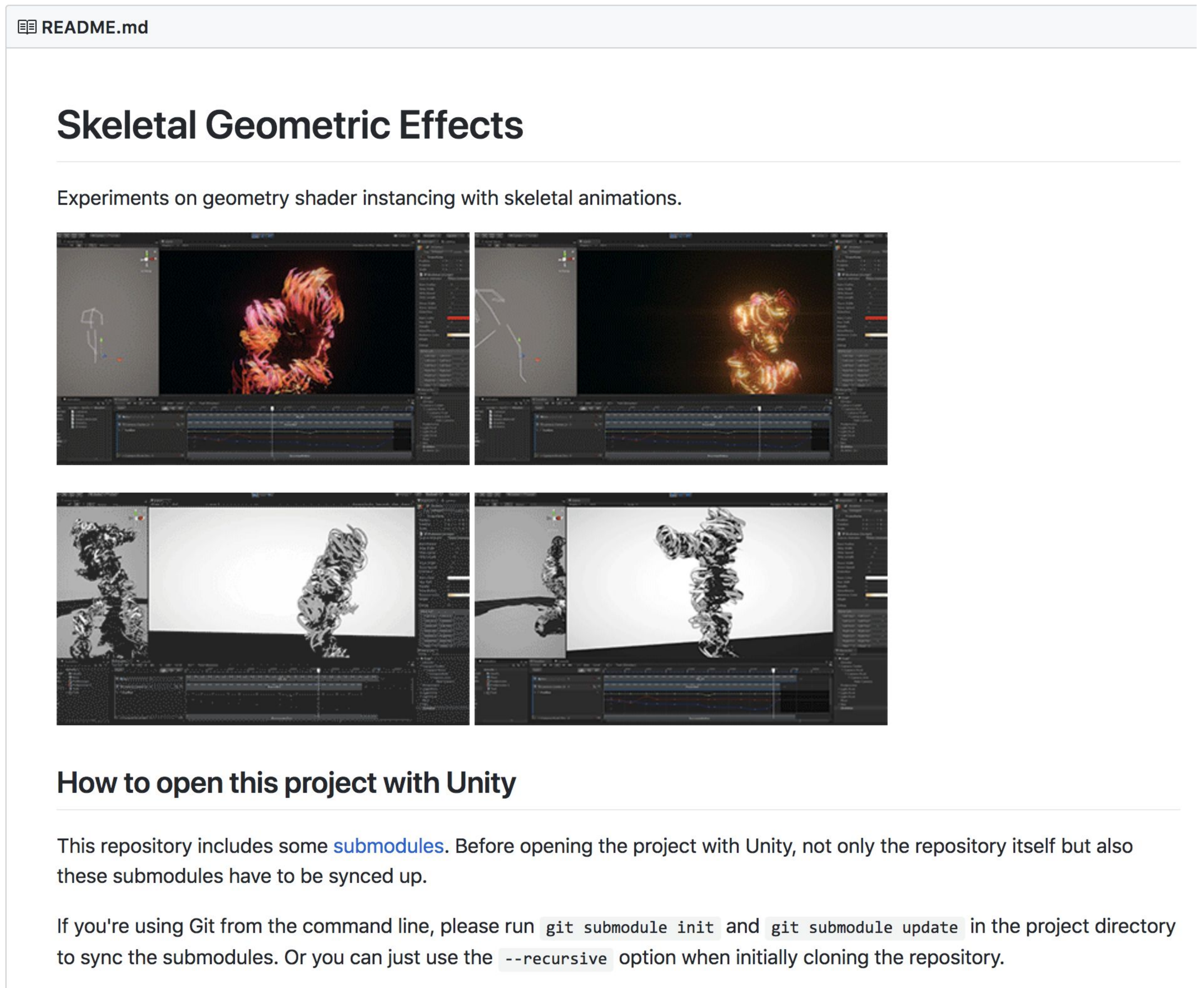
- "...because it's crap"
    - *pssst:* all code is crap, so stop worrying!
    - Less-than-ideal code online is miles better than no code
    - Especially if applying for junior; it’s *expected* that code’s not great

”I don't want to put my code online because...”

- "...because someone will steal/copy it"
  - Sorry, no one’s gonna copy/steal your code
  - Most likely no one’s gonna *see* it unless you point them to it
  - Sorry, your code is likely not that great either

Portfolio: code

- If you have >3 projects online:
- Put nice descriptions / screenshots on “best” ones



README.md

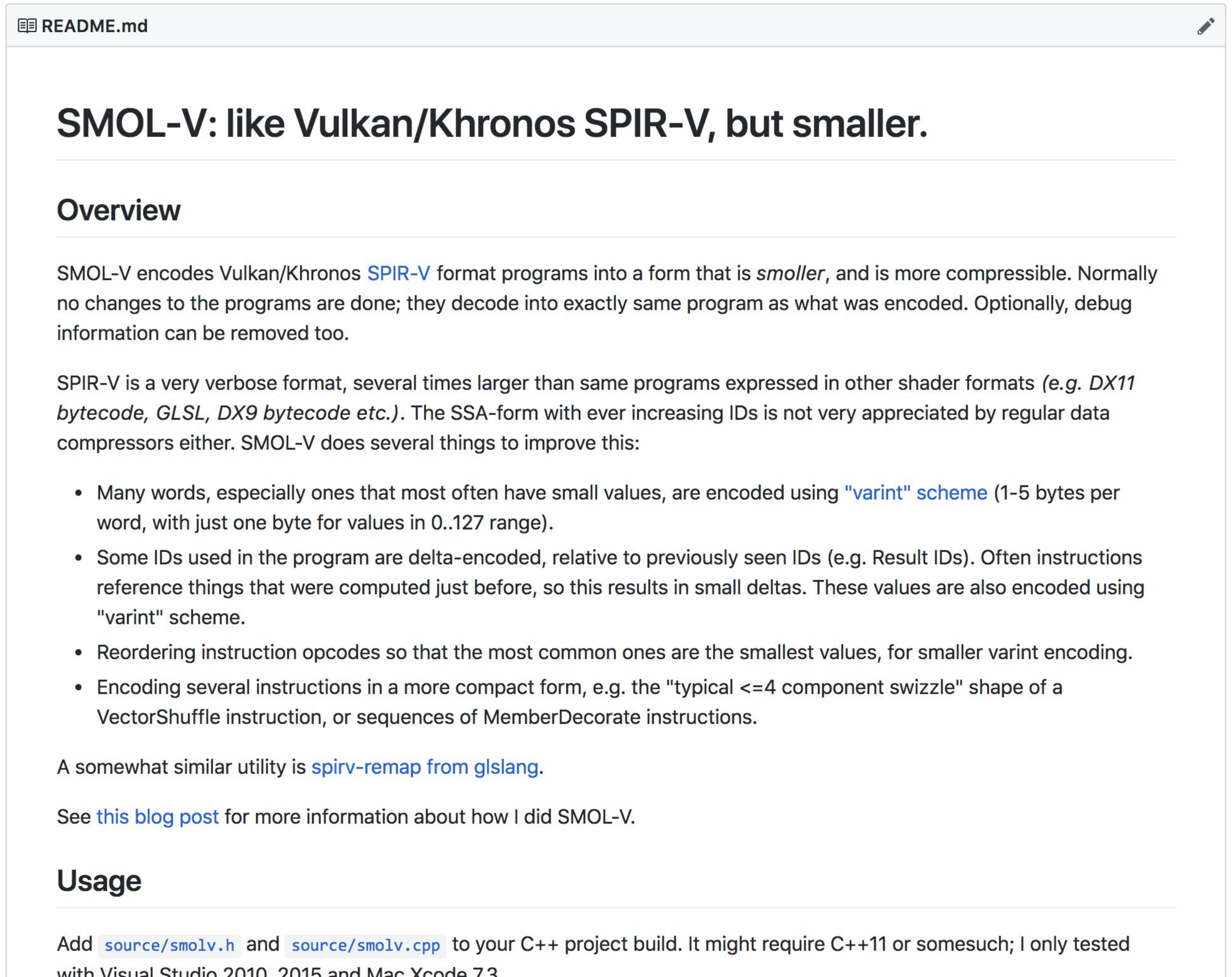
Skeletal Geometric Effects

Experiments on geometry shader instancing with skeletal animations.

How to open this project with Unity

This repository includes some submodules. Before opening the project with Unity, not only the repository itself but also these submodules have to be synced up.

If you're using Git from the command line, please run `git submodule init` and `git submodule update` in the project directory to sync the submodules. Or you can just use the `--recursive` option when initially cloning the repository.



README.md

SMOL-V: like Vulkan/Khronos SPIR-V, but smaller.

Overview

SMOL-V encodes Vulkan/Khronos SPIR-V format programs into a form that is *smoller*, and is more compressible. Normally no changes to the programs are done; they decode into exactly same program as what was encoded. Optionally, debug information can be removed too.

SPIR-V is a very verbose format, several times larger than same programs expressed in other shader formats *(e.g. DX11 bytecode, GLSL, DX9 bytecode etc.)*. The SSA-form with ever increasing IDs is not very appreciated by regular data compressors either. SMOL-V does several things to improve this:

- Many words, especially ones that most often have small values, are encoded using "varint" scheme (1-5 bytes per word, with just one byte for values in 0..127 range).
- Some IDs used in the program are delta-encoded, relative to previously seen IDs (e.g. Result IDs). Often instructions reference things that were computed just before, so this results in small deltas. These values are also encoded using "varint" scheme.
- Reordering instruction opcodes so that the most common ones are the smallest values, for smaller varint encoding.
- Encoding several instructions in a more compact form, e.g. the "typical <=4 component swizzle" shape of a VectorShuffle instruction, or sequences of MemberDecorate instructions.

A somewhat similar utility is spirv-remap from glslang.

See this blog post for more information about how I did SMOL-V.

Usage

Add `source/smolv.h` and `source/smolv.cpp` to your C++ project build. It might require C++11 or somesuch; I only tested with Visual Studio 2010, 2015 and Mac Xcode 7.3

Portfolio: blog / technical social media

- Make an account
  - wordpress / medium / tumblr / jekyll / hugo / twitter
- Write about your coding adventures
- Do that repeatedly
- Do that in English
  - No point in limiting your audience to the 7 LT people

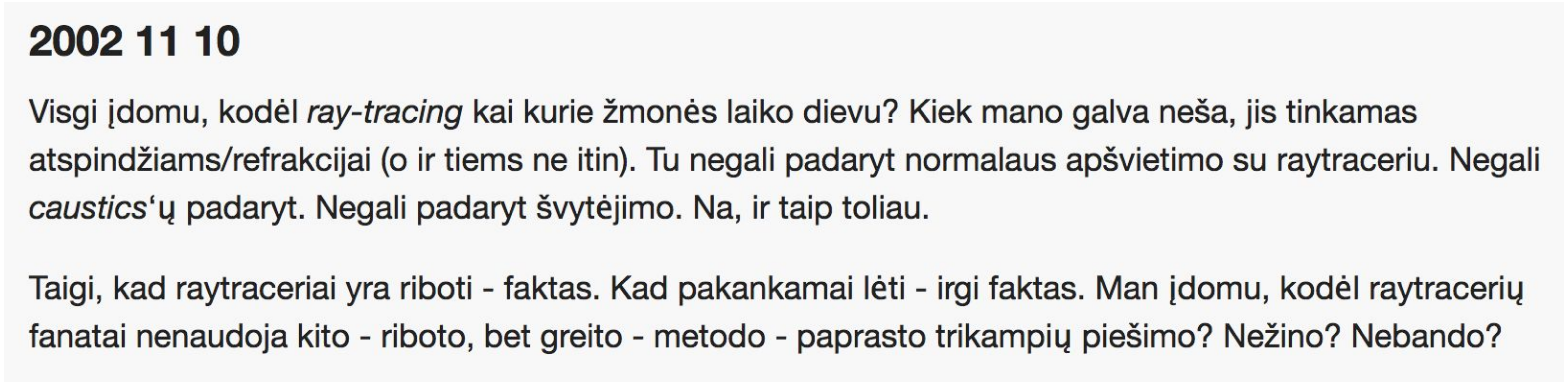
Portfolio: blog / technical social media

- "but I don't have anything interesting to say"
  - *pssst:* people with 20 years experience feel the same
  - Your blog is not there to win a Turing award
  - Nor to convey excellent/useful information (yet)
  - It’s there to show: interest, work put in, learning

unity

Portfolio: blog / technical social media

- Here's start of *my* blog in 2002, with all mistakes:
  - In Lithuanian… took until 2005 to switch to English
  - Where I'm talking nonsense about raytracing, ha, ha, ha
  - All that stupidity didn't stop me for next 16 years



2002 11 10

Visgi įdomu, kodėl *ray-tracing* kai kurie žmonės laiko dievu? Kiek mano galva neša, jis tinkamas atspindžiams/refrakcijai (o ir tiems ne itin). Tu negali padaryt normalaus apšvietimo su raytraceriu. Negali *caustics*'ų padaryt. Negali padaryt švytėjimo. Na, ir taip toliau.

Taigi, kad raytraceriai yra riboti - faktas. Kad pakankamai lėti - irgi faktas. Man įdomu, kodėl raytracerių fanatai nenaudoja kito - riboto, bet greito - metodo - paprasto trikampių piešimo? Nežino? Nebando?

Portfolio: blog / technical social media

- Actually good blog example
  - https://megascopsdev.tumblr.com/
  - Daily adventures in coding up a small DX12 engine
    - Including all the “boring/trivial” bits
  - Fun fact: author (Elizabeth) works at Unity now, on ECS fancy

Programming Test

Programming Test

- Could be questionnaire, could be a task
    - Depends on position too,
    - e.g. graphics programmers get math/geometry/algebra
- Time: couple hours to couple evenings
    - Again, introduces bias against people who can’t afford this…
- Updates to your solution are often possible

unity

Programming Test

- Things evaluated, often in this order:
  - Correctness,
  - Clarity (can code be read & understood?),
  - Good practices (tests, structure, …)
  - Performance

- Things *also* evaluated:
  - “Show off” of tricks, over-engineering? (usually: don’t do that)
  - How do you manage time? (submit on last day, …)
  - How do you communicate? (clarify vs guessing, …)
  - Overly apologetic, or overly confident?
- Things not directly in your code *do* tell quite a lot about you!

- Do not copy-paste solutions from the internet
  - The interviewers are not stupid, they will notice
  - Often problems are subtly different from solutions on the net
    - Not only you copy-pasted, but also copy-pasted the wrong answer!

Phone & On-site Interview

unity

Often in two parts: Technical & Cultural

- Technical part fairly obvious
    - Less googling & more knowledge that you actually have
- Cultural part, mostly common sense
    - Show interest, be respectful, …
    - Varies a lot by company/location/position too

Tips

- Better to say “I don’t know” than trying to wing it.
  - ...this is not University :)
- Don’t go too negative on yourself
  - e.g. don’t start with “I'm probably talking nonsense here”
- Don’t be overly confident!
  - “Why don’t you *just*” → the “just” part often means “6 months of work”
  - “Simply *multithread* that” → threading is *never* simple
- A lot of Uni knowledge is not-that-useful
  - OOP, Design Patterns, UML, red-black trees, …

unity

Onsite Interview!

- If you get there, company should pay all expenses
  - Travel, hotel, food etc.
  - Your expense is time. 1-3 days, which is quite a lot.
    - OTOH you get experience, see company, meet people
- Rules again are mostly common sense
- Like an extended version of previous ones
- More “feel” in how you interact, talk, discuss

unity

What if you get rejected?

- It can lead to great things!
  - You got experience, learned something
  - Try elsewhere,
  - ...etc.
- I failed interview with AAA studio in 2005
  - One of best failures I ever did :)

unity

Ask me 5 or more questions!

unity