The Minefield Beyond Algorithms

Nandakumar Edamana

2025-02-05

What's Wrong With This Picture?

What’s Wrong With This Picture?

We’ll come back to it later.

What’s Wrong With This Code? (1)

Let’s start with something very simple:

```
double x, y;
scanf("%lf %lf", &x, &y);
printf("x / y = %f\n", x / y);
```

What's Wrong With This Code? (1)

Nothing, at least from a crash-perspective.

What's Wrong With This Code? (1)

Nothing, at least from a crash-perspective.

Believe me, $y = 0$ won't result in a crash.

What's Wrong With This Code? (1)

Nothing, at least from a crash-perspective.

Believe me, $y = 0$ won't result in a crash.

But you clearly remember it crashing the other day, right?
What's different this time?

What’s Wrong With This Code? (2)

```
#include <stdio.h>

int main()
{
    char *buf = malloc(1024);
    ...
```

What's Wrong With This Code? (2)

Pointer truncation. Here's the fix:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char *buf = malloc(1024);
    ...
```

What's Wrong With This Code? (2)

Pointer truncation. Here's the fix:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char *buf = malloc(1024);
    ...
```

Can happen even after including `stdlib.h`, if you pass the pointer to an undeclared custom function.

What's Wrong With This Code? (2)

The truncated value can be:

1. An invalid pointer

What's Wrong With This Code? (2)

The truncated value can be:

1. An invalid pointer
2. Same as the original pointer (MSB $=$ 0x0000)

What’s Wrong With This Code? (2)

The truncated value can be:

1. An invalid pointer
2. Same as the original pointer (MSB = 0x0000)
3. Different, pointing to a location owned by the process

What's Wrong With This Code? (3)

Now what?

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char *buf = malloc(1024);
    buf[0] = 0;
    ...
```

What's Wrong With This Code? (3)

Didn't check the return value of `malloc()`. Duh.

What's Wrong With This Code? (4)

So this program is perfectly safe?

```
#include <stdio.h>
#include <stdlib.h>

const size_t GB = 1024 * 1024 * 1024;

int main()
{
    char *buf = malloc(8 * GB);
    if(buf == NULL) { /* Say error and exit */ }

    /* Use buf */
    ...
```

What’s Wrong With This Code? (4)

`malloc()` can fail in future.

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)
- You didn't check how much is free; you just asked for 8 GB

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)
- You didn't check how much is free; you just asked for 8 GB
- But malloc() returned a valid pointer

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)
- You didn't check how much is free; you just asked for 8 GB
- But malloc() returned a valid pointer
- You used nearly 6 GB; so far so good.

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)
- You didn't check how much is free; you just asked for 8 GB
- But malloc() returned a valid pointer
- You used nearly 6 GB; so far so good.
- You started using the remaining, and after a while...

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)
- You didn't check how much is free; you just asked for 8 GB
- But malloc() returned a valid pointer
- You used nearly 6 GB; so far so good.
- You started using the remaining, and after a while...
- Crash.

The Inevitable Doom

Scenario:

- You have 10 GB memory (RAM + swap)
- You have 6 GB memory free (RAM + swap)
- You didn't check how much is free; you just asked for 8 GB
- But malloc() returned a valid pointer
- You used nearly 6 GB; so far so good.
- You started using the remaining, and after a while...
- Crash.

You were being a good citizen. Why did the OS betray you?

Memory Overcommit

The Linux kernel overcommit handling modes:

- 0 - Heuristic overcommit. Ensures a seriously wild allocation fails while allowing **overcommit to reduce swap usage**.

Memory Overcommit

The Linux kernel overcommit handling modes:

- 0 - Heuristic overcommit. Ensures a seriously wild allocation fails while allowing **overcommit to reduce swap usage**.
- 1 - Always overcommit. Appropriate for some scientific applications. Classic example is **code using sparse arrays**.

Memory Overcommit

The Linux kernel overcommit handling modes:

- 0 - Heuristic overcommit. Ensures a seriously wild allocation fails while allowing **overcommit to reduce swap usage**.
- 1 - Always overcommit. Appropriate for some scientific applications. Classic example is **code using sparse arrays**.
- 2 - Don't overcommit. For applications that want to guarantee their memory allocations will be available in the future without having to initialize every page.

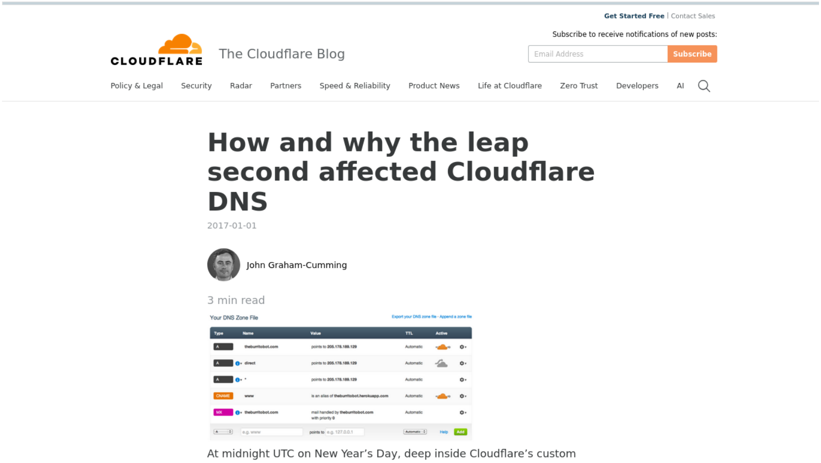
See the doc for more details.

Memory Overcommit

From https://www.kernel.org/doc/Documentation/vm/overcommit-accounting:

- The overcommit policy is set via the sysctl `vm.overcommit_memory`
- The default is 0 (heuristic overcommit)

Time Flows Backwards. . .



Get Started Free | Contact Sales

Subscribe to receive notifications of new posts:

How and why the leap second affected Cloudflare DNS

2017-01-01

John Graham-Cumming

3 min read

Your DNS Zone File

Type	Name	Value	TTL	Active
A	theburritobot.com	points to 205.178.189.129	Automatic	
A	direct	points to 205.178.189.129	Automatic	
A	*	points to 205.178.189.129	Automatic	
CNAME	www	is an alias of theburritobot.herokuapp.com	Automatic	
MX	theburritobot.com	mail handled by theburritobot.com with priority 0	Automatic	
A	e.g. www	points to e.g. 127.0.0.1	Automatic	Help Add

At midnight UTC on New Year’s Day, deep inside Cloudflare’s custom

Root cause: Go’s `time.Now()` was not monotonic.

The Limitless Minefield

- Date/time: overflow, timezone, daylight saving, etc.

The Limitless Minefield

- Date/time: overflow, timezone, daylight saving, etc.
- Resulting type of division

The Limitless Minefield

- Date/time: overflow, timezone, daylight saving, etc.
- Resulting type of division
- Floating-point comparison

The Limitless Minefield

- Date/time: overflow, timezone, daylight saving, etc.
- Resulting type of division
- Floating-point comparison
- Issue that Java's Binary Search implementation had

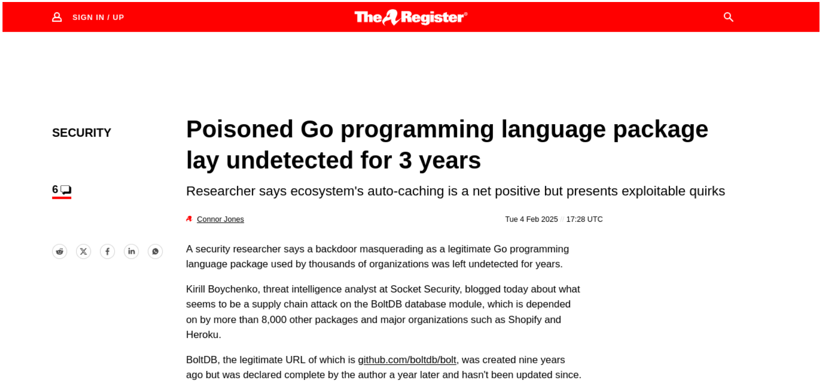
The Limitless Minefield

- Date/time: overflow, timezone, daylight saving, etc.
- Resulting type of division
- Floating-point comparison
- Issue that Java's Binary Search implementation had
- Use of unsigned integers

The Limitless Minefield

- Date/time: overflow, timezone, daylight saving, etc.
- Resulting type of division
- Floating-point comparison
- Issue that Java's Binary Search implementation had
- Use of unsigned integers
- Supply-chain attacks (npm, PyPI, etc.)

Supply-chain Attacks



SECURITY

6

Poisoned Go programming language package lay undetected for 3 years

Researcher says ecosystem's auto-caching is a net positive but presents exploitable quirks

Connor Jones   Tue 4 Feb 2025 // 17:28 UTC

A security researcher says a backdoor masquerading as a legitimate Go programming language package used by thousands of organizations was left undetected for years.

Kirill Boychenko, threat intelligence analyst at Socket Security, blogged today about what seems to be a supply chain attack on the BoltDB database module, which is depended on by more than 8,000 other packages and major organizations such as Shopify and Heroku.

BoltDB, the legitimate URL of which is github.com/boltdb/bolt, was created nine years ago but was declared complete by the author a year later and hasn't been updated since.

Figure 1: A Latest Example

Supply-chain Attacks

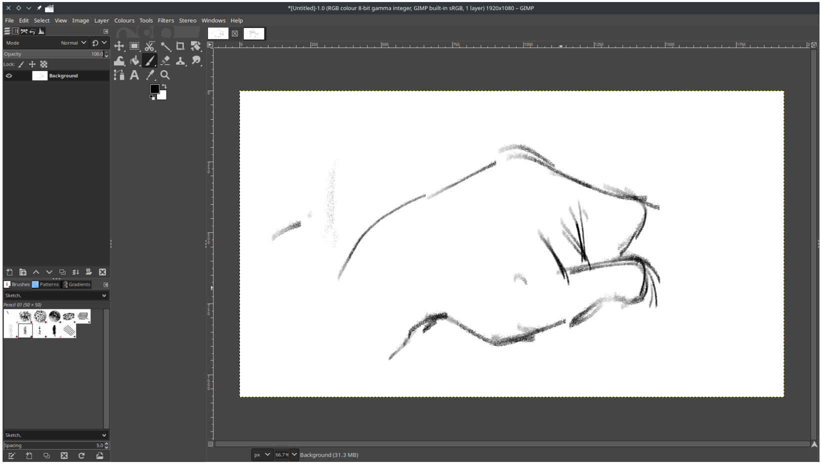
Vara doesn’t have any extraneous dependencies

Supply-chain Attacks

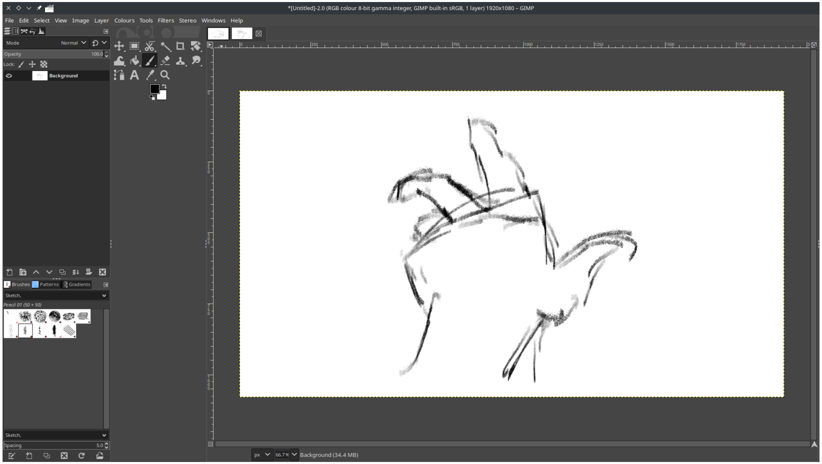
Vara doesn't have any extraneous dependencies

Wait, I didn't mention what Vara is.

Wanted to Sketch. . . (detour)


*[Untitled]-1.0 (RGB colour 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 1920x1080 – GIMP
File Edit Select View Image Layer Colours Tools Filters Stereo Windows Help
Mode Normal
Opacity 100.0
Lock:
Background
Brushes Patterns Gradients
Sketch,
Pencil 01 (50 × 50)
Sketch,
Spacing 5.0
px 66.7%
Background (31.3 MB)

Wanted to Sketch. . . (detour)


*[Untitled]-2.0 (RGB colour 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 1920x1080 – GIMP
File Edit Select View Image Layer Colours Tools Filters Stereo Windows Help
Mode Normal
Opacity 100.0
Lock:
Background
Brushes Patterns Gradients
Sketch,
Pencil 01 (50 × 50)
Sketch,
Spacing 5.0
px 66.7% Background (34.4 MB)

Wanted to Sketch. . . (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita

Wanted to Sketch. . . (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita – great, but not small or simple

Wanted to Sketch. . . (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita – great, but not small or simple
- GIMP

Wanted to Sketch. . . (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita – great, but not small or simple
- GIMP – good for editing, not for drawing

Wanted to Sketch... (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita – great, but not small or simple
- GIMP – good for editing, not for drawing
- ...

Wanted to Sketch. . . (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita – great, but not small or simple
- GIMP – good for editing, not for drawing
- . . .

Easiest solution: stop sketching.

Wanted to Sketch. . . (detour)

Requirement: a libre drawing application for GNU/Linux that is small, simple, usable, with pressure-sensitive brushes.

- Krita – great, but not small or simple
- GIMP – good for editing, not for drawing
- . . .

Easiest solution: stop sketching.

Second option: develop one.

a.out (detour)

Day 1. Didn't even bother to rename the binary. . .

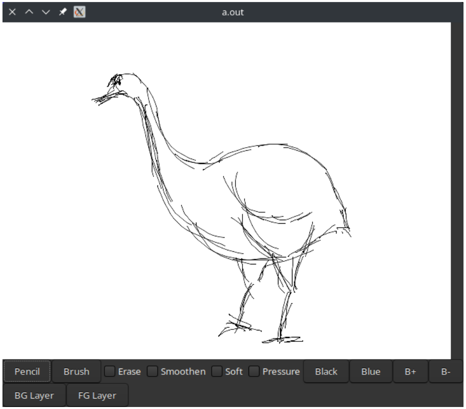


Figure 2: 2023-05-11

a.out (detour)

Next day. . .

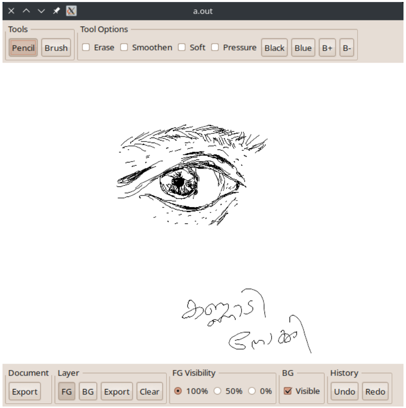


Figure 3: 2023-05-12

Vara (detour)

Next month. . .

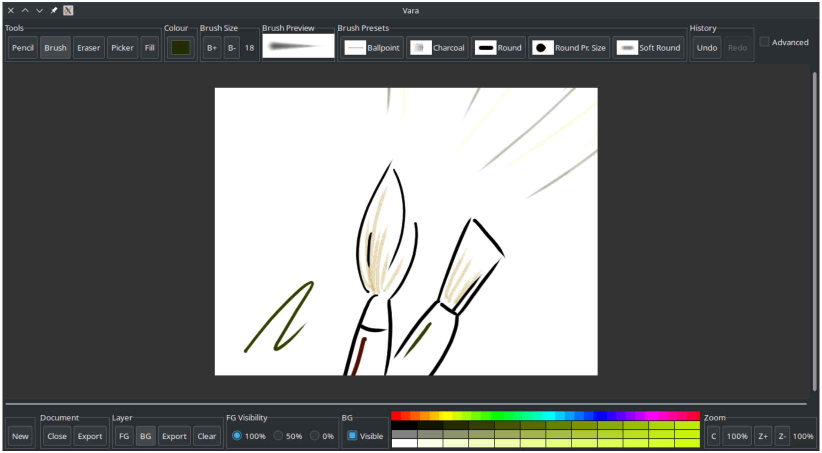
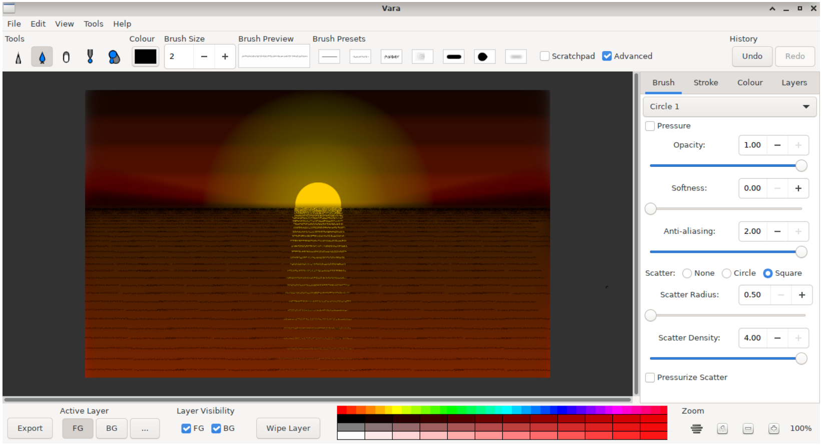


Figure 4: 2023-06-14

Vara (detour)

Today. . .



Brush Size: 2

[✓ Advanced]

Opacity: 1.00

Softness: 0.00

Anti-aliasing: 2.00

Scatter: None Circle [✓ Square]

Scatter Radius: 0.50

Scatter Density: 4.00

Figure 5: 2024-08-23

Vara (detour)

Vara has:

- Pressure-sensitive brushes with stroke smoothing
- Layers, Undo/Redo, HSL
- Brush presets, Quick palette, Keyboard shortcuts, Zooming
- Save and open XCF, export PNG
- Linear RGB internals and Gamma Correction

All in 11k lines of C, core processing done without any third-party libraries.

Vara (detour)

- Released on Flathub, Snap Store, etc.
- Free/Open Source under GNU GPL v3.

Bootstrapping the Logo (detour)

Not creative, I agree, but at least it's procedural. . .

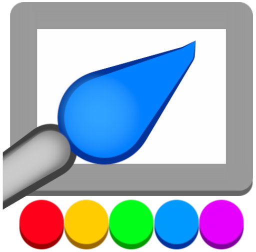


Figure 6: Logo of Vara

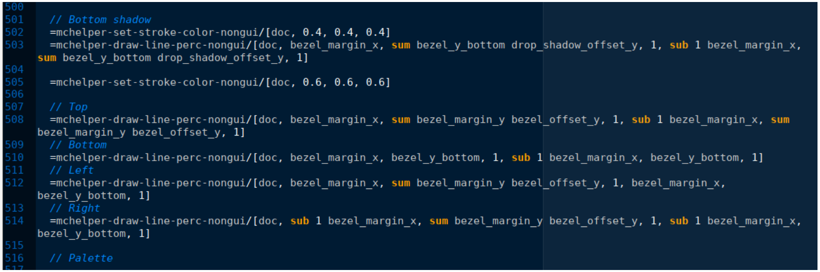
Bootstrapping the Logo (detour)

Not creative, I agree, but at least it's procedural. . .

Figure 6: Logo of Vara

. . . meaning it in itself is a test.

Bootstrapping the Logo (detour)



```

  // Bottom shadow
  =mchelper-set-stroke-color-nongui/[doc, 0.4, 0.4, 0.4]
  =mchelper-draw-line-perc-nongui/[doc, bezel_margin_x, sum bezel_y_bottom drop_shadow_offset_y, 1, sub 1 bezel_margin_x,
sum bezel_y_bottom drop_shadow_offset_y, 1]

  =mchelper-set-stroke-color-nongui/[doc, 0.6, 0.6, 0.6]

  // Top
  =mchelper-draw-line-perc-nongui/[doc, bezel_margin_x, sum bezel_margin_y bezel_offset_y, 1, sub 1 bezel_margin_x, sum
bezel_margin_y bezel_offset_y, 1]
  // Bottom
  =mchelper-draw-line-perc-nongui/[doc, bezel_margin_x, bezel_y_bottom, 1, sub 1 bezel_margin_x, bezel_y_bottom, 1]
  // Left
  =mchelper-draw-line-perc-nongui/[doc, bezel_margin_x, sum bezel_margin_y bezel_offset_y, 1, bezel_margin_x,
bezel_y_bottom, 1]
  // Right
  =mchelper-draw-line-perc-nongui/[doc, sub 1 bezel_margin_x, sum bezel_margin_y bezel_offset_y, 1, sub 1 bezel_margin_x,
bezel_y_bottom, 1]

  // Palette
```

Figure 7: A small portion from the code that draws the logo

Visual Tests (detour)

Apart from `vara --test-nongui`:

- `vara --test-sunset`
- `vara --test`

Back on Track...

But that's not the point.

Not So Easy

Even for a simple drawing application, you need:

- Anti-aliasing
- Stroke smoothing
- Premultiplied alpha
- Gamma correction
- Multiple color spaces/representations

Not So Easy

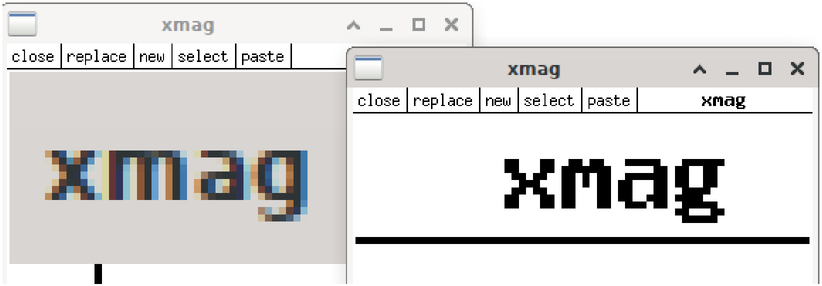


Figure 8: Anti-aliasing (left: with, right: without)

Not So Easy

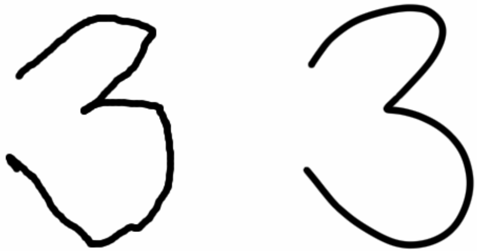


Figure 9: Stroke Smoothing

What Was Wrong With This Picture?

Enter Gamma

Enter Gamma

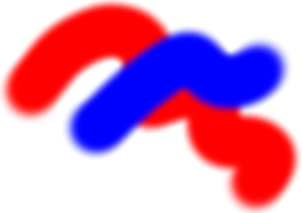


Figure 10: Improper

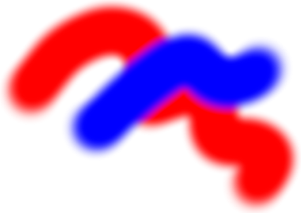


Figure 11: Proper

Gamma

Disclaimer: my explanation could be off.

Gamma

Disclaimer: my explanation could be off.

- Camera sensors, image processing engines, etc. use linear color

Gamma

Disclaimer: my explanation could be off.

- Camera sensors, image processing engines, etc. use linear color
- Linear: double the value, double the intensity

Gamma

Disclaimer: my explanation could be off.

- Camera sensors, image processing engines, etc. use linear color
- Linear: double the value, double the intensity
- Human vision has a non-linear response

Gamma

Disclaimer: my explanation could be off.

- Camera sensors, image processing engines, etc. use linear color
- Linear: double the value, double the intensity
- Human vision has a non-linear response
- Store and transmit the images with non-linear encoding for efficient use of bits

Gamma

Disclaimer: my explanation could be off.

- Camera sensors, image processing engines, etc. use linear color
- Linear: double the value, double the intensity
- Human vision has a non-linear response
- Store and transmit the images with non-linear encoding for efficient use of bits
- Multiply with Gamma

Gamma

Disclaimer: my explanation could be off.

- Camera sensors, image processing engines, etc. use linear color
- Linear: double the value, double the intensity
- Human vision has a non-linear response
- Store and transmit the images with non-linear encoding for efficient use of bits
- Multiply with Gamma
- But this has to be undone before processing

Basics: Chroma and Alpha

- (0, 0, 1, **0**) - Fully transparent blue
- (1, 1, 0, **0.5**) - Half-transparent yellow
- (0, 1, 1, **1**) - Fully opaque cyan

(In case you care, this is straight alpha, not premultiplied.)

Alpha Compositing

Consider pixels A (top layer) and B (bottom layer). A has an alpha α.

$C = \alpha A + (1 - \alpha)B$

Alpha Compositing

Consider pixels A (top layer) and B (bottom layer). A has an alpha α.

$C = \alpha A + (1 - \alpha)B$

Remember: If directly read from the input, you have A^γ, not A.

Alpha Compositing

Consider pixels A (top layer) and B (bottom layer). A has an alpha α.

$C = \alpha A + (1 - \alpha)B$

Remember: If directly read from the input, you have A^γ, not A.

Now you know why the overlapping region was darker without proper gamma processing.

Gamma

From https://blog.johnnovak.net/2016/09/21/what-every-coder-should-know-about-gamma/:

> *The fact that most computer graphics textbooks don't explicitly mention the importance of correct gamma handling, or discuss it in practical terms, does not help matters at all...*

Premultiplied Alpha

- Premultiplied alpha was “rejected in the design of PNG", according to libpng.org
- GIMP XCF does not use premultiplied alpha
- libcairo uses premultiplied alpha
- Premultiplied alpha is necessary at least internally for correct compositing

ngg (detour)

Vara is not a side project;

ngg (detour)

Vara is not a side project; it's a side-side-project.

ngg (detour)

Vara is not a side project; it’s a side-side-project.

Vara is just an example for how useful ngg is.

ngg (detour)

Vara is not a side project; it's a side-side-project.

Vara is just an example for how useful ngg is.

Vara is written in ngg. If written directly in C, I'd still be chasing segfaults instead of coding up the actual painting logic.

ngg (detour)

- Strongly and statically typed
- Multi-paradigm, mainly OOP
- Semi-automatic memory management, Static reflection, Templates, etc.
- Tight integration with C
- Generated code: modular, maintainable, near-zero overhead
- Compiles to C (mature), Go, JavaScript, Assembly, etc. (WIP)
- Self-hosted transpiler
- In active development since 2019

ngg: Materialistic Stats (detour)

Example: ngg generates the 419kB C source code of Vara from 222kB of ngg source.

ngg: Materialistic Stats (detour)

Example: ngg generates the 419kB C source code of Vara from 222kB of ngg source.

Read: ngg saved me 200k keystrokes and hours of insane debugging.

ngg: Materialistic Stats (detour)

Example: ngg generates the 419kB C source code of Vara from 222kB of ngg source.

Read: ngg saved me 200k keystrokes and hours of insane debugging.

Read: I waste a lot of time developing things to develop things instead of developing the things I should be developing.

ngg Example (detour)

ngg source:

```
class Person takes name own mstring;
```

ngg Example (detour)

.c output:

```
typedef struct Person {
    char * name;
} Person;

void person_construct(Person *this, char * name)
{
    this->name = name;
}

void person_destruct(Person *this)
{
    if(this->name) {
        free(this->name);
    }
}
```

ngg Example (detour)

.h output:

```
void person_construct(Person *this, char * name);
void person_destruct(Person *this);
```

ngg (detour)

The point - ngg was started to deal with pitfalls. Now it has:

- Explicit `nullable`
- Some notion of ownership (not as robust as Rust)
- Better type safety (compared to C)

. . . and more.

Fruit for Thought

- Undefined behaviour
- Unspecified behaviour
- Implementation-defined behaviour

Fruit for Thought

Is there a way to reliably determine if a piece of data has been written to the disk?

Discussion