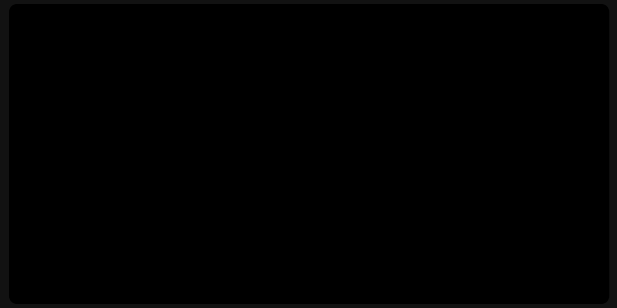
UN PIXEL PUIS UN AUTRE

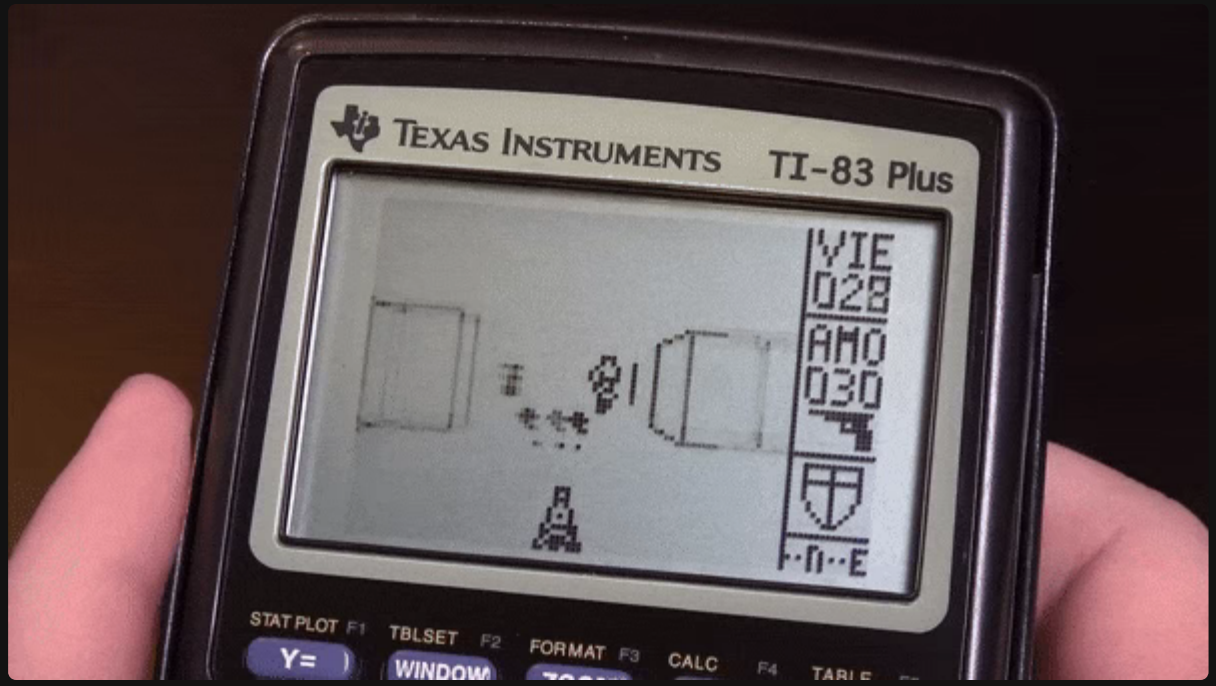
Comment porter Doom sur kindle

DOOM

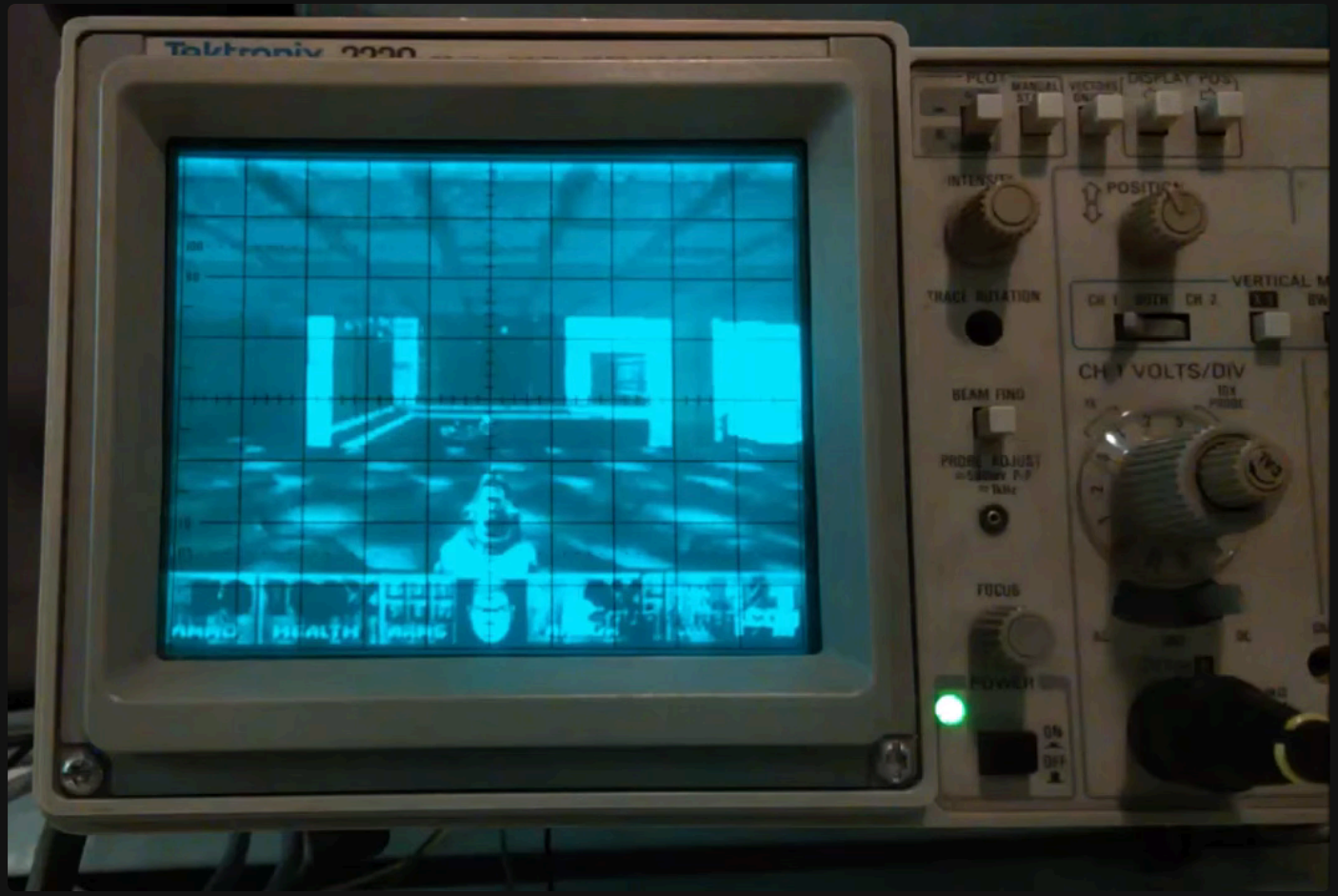


Timeline

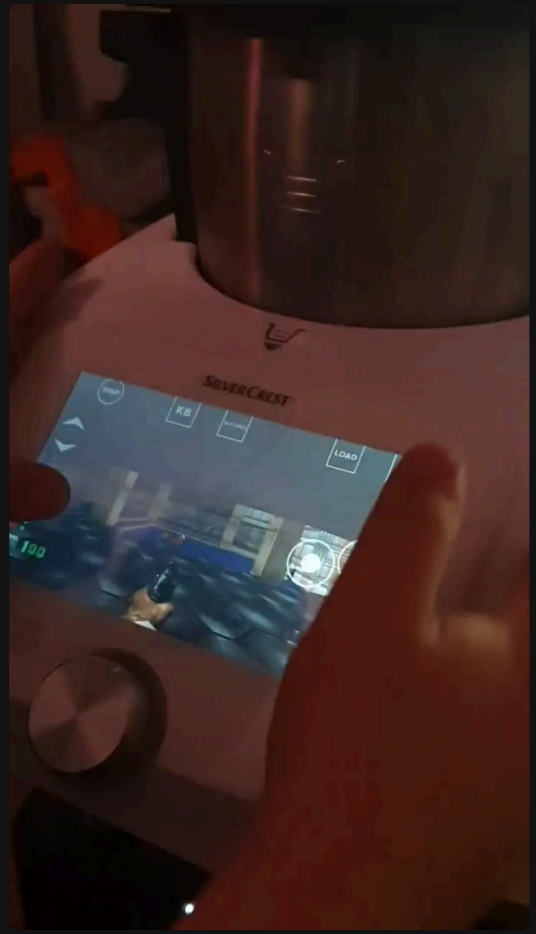
Timeline stuff here



LGR - Running Doom on a Calculator! TI-83 Plus

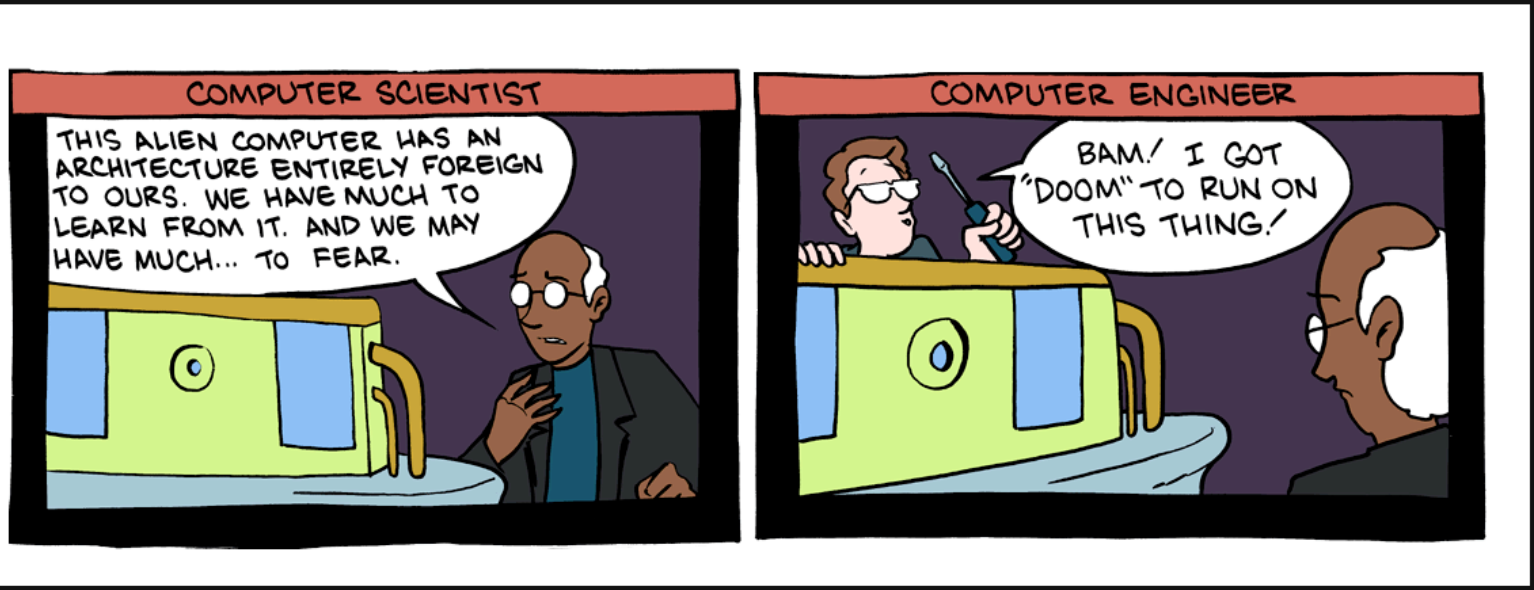


MrSlehofer - Doom on an Oscilloscope (Tektronix 2220)



Siphonay - Made a Thermomix clone run Doom with a friend

@Foon - You were trying to get pregnant, right?


COMPUTER SCIENTIST
THIS ALIEN COMPUTER HAS AN ARCHITECTURE ENTIRELY FOREIGN TO OURS. WE HAVE MUCH TO LEARN FROM IT. AND WE MAY HAVE MUCH... TO FEAR.
COMPUTER ENGINEER
BAM! I GOT "DOOM" TO RUN ON THIS THING!

id-Software/DOOM

id-mikerubits Add GPL information	a77dfb9 · 10 months ago	4 Commits
ipx	Adding the release of the DOOM IPX driver.	13 years ago
linuxdoom-1.10	Add GPL information	10 months ago
sersrc	The source for the DOOM serial / model driver.	13 years ago
sndserv	The DOOM sources as originally released on December 2...	13 years ago
LICENSE.TXT	Add GPL information	10 months ago
README.TXT	Add GPL information	10 months ago

README GPL-2.0 license

```
Here it is, at long last.  The DOOM source code is released for your
non-profit use.  You still need real DOOM data to work with this code.
If you don't actually own a real copy of one of the DOOMs, you should
still be able to find them at software stores.

Many thanks to Bernd Kreimeier for taking the time to clean up the
project and make sure that it actually works.  Projects tends to rot if
you leave it alone for a few years, and it takes effort for someone to
deal with it again.
```

About

DOOM Open Source Release

- Readme
- GPL-2.0 license
- Activity
- Custom properties
- **14.4k** stars
- **418** watching
- **2.3k** forks

Report repository

Releases

No releases published

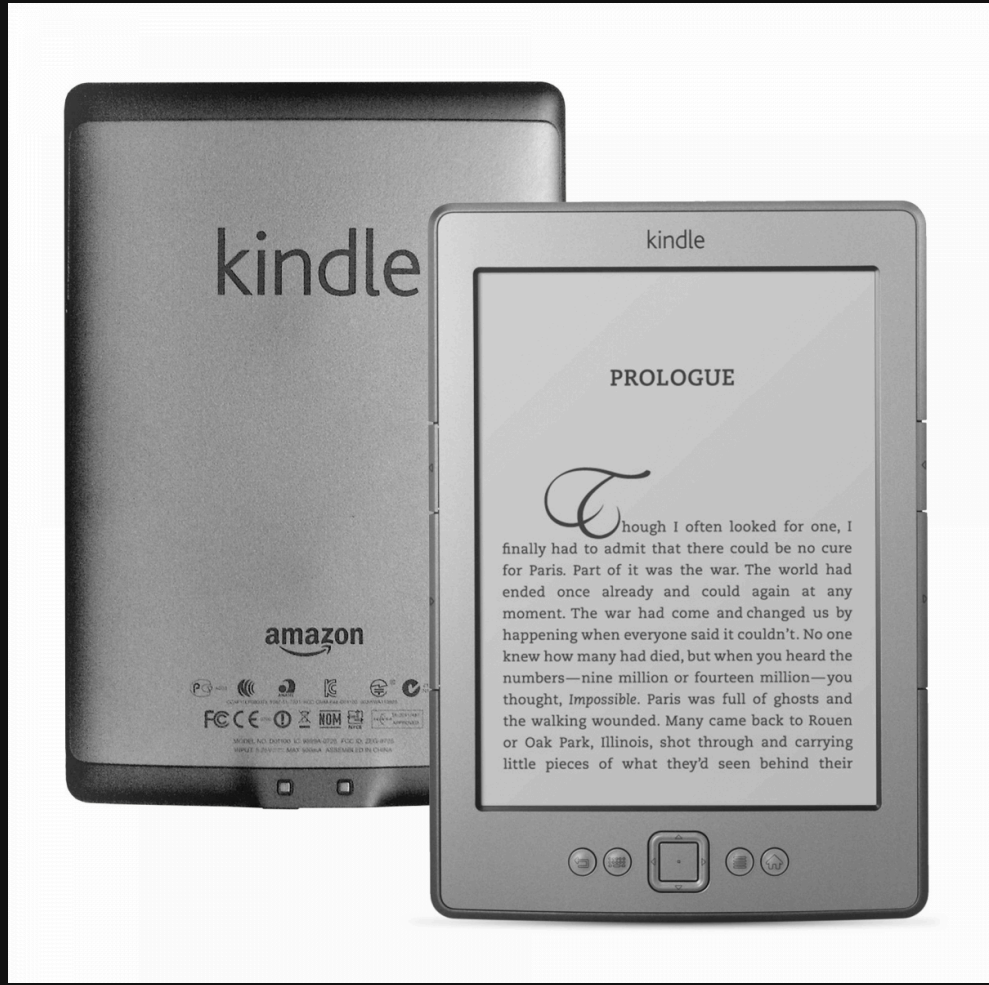
Packages

No packages published

Languages

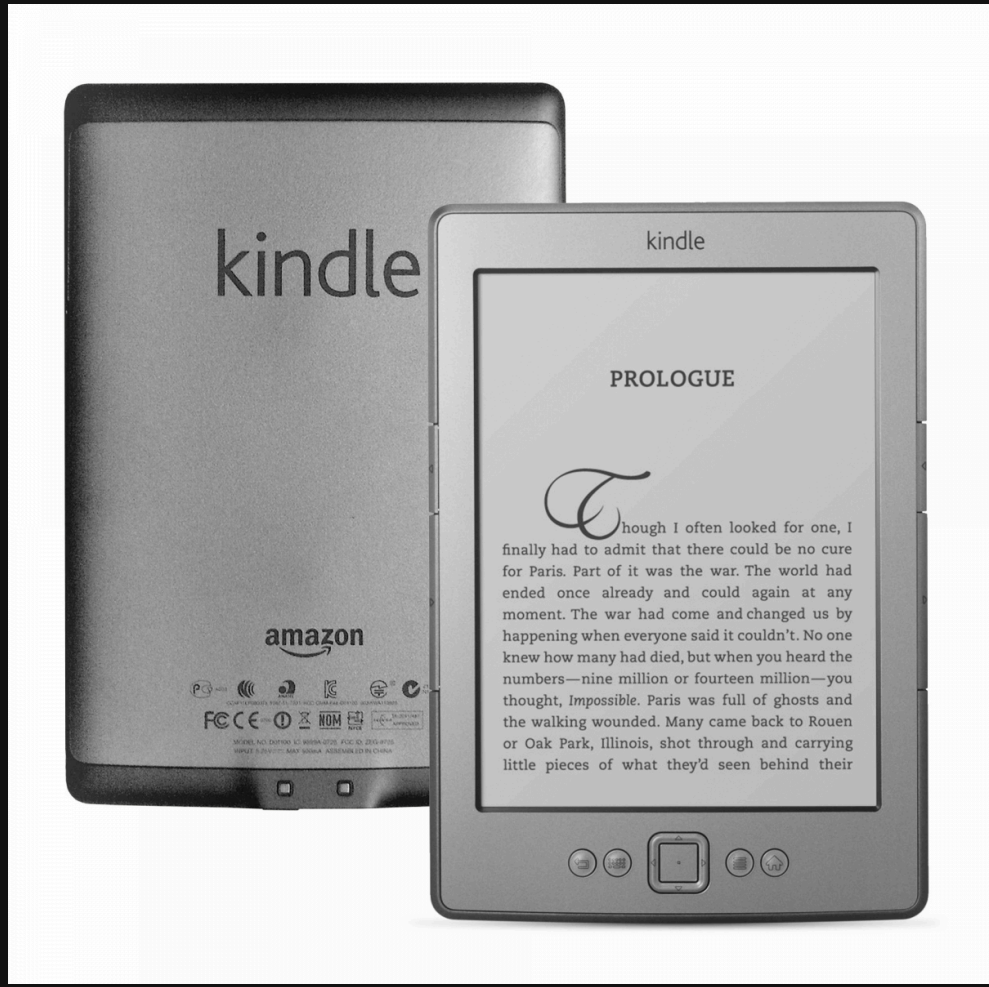
- **C++** 96.0%
- **C** 3.8%
- **Makefile** 0.2%

Kindle 4th Gen



Kindle 4th Gen

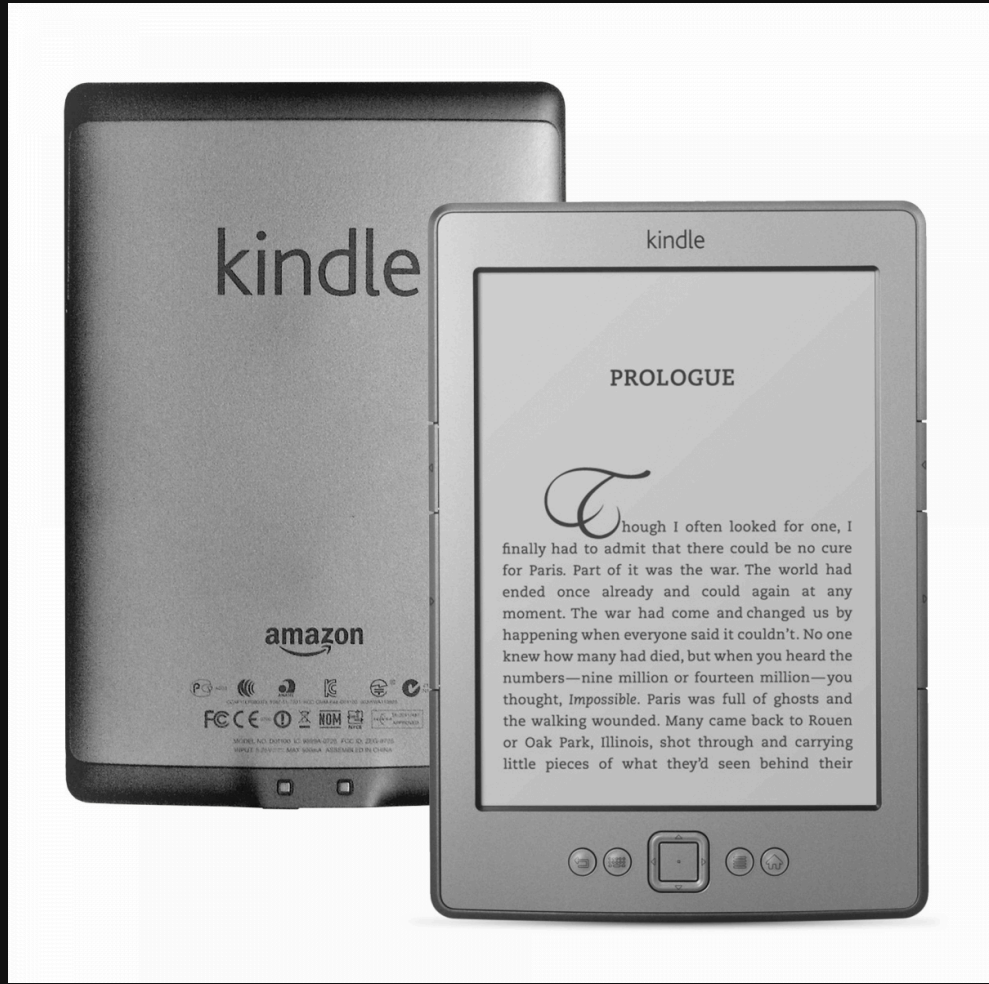
(qui date un peu…)

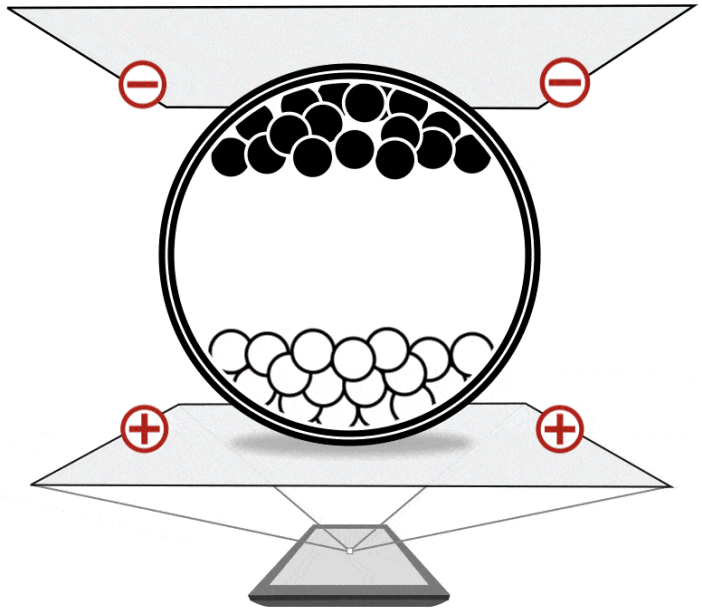


Kindle 4th Gen

(qui date un peu...)

Release	`September 2011`
Display	`6"` `600×800p` `8 bit Greyscale`
OS	`Linux (Amazon)`
Memory	`256 MB DDR`
CPU	`ARM Cortex-A8` `800MHz` `Freescale i.MX508`





Two Particle Ink System - E Ink Carta™

Jailbreak TL;DR

Step I : Se faire passer pour un développeur Amazon

Jailbreak TL;DR

Step I : Se faire passer pour un développeur Amazon

`Kindle4NTHacking.tar.gz` Fake Dev Key + Diagnostic mode + Script ✨

Jailbreak TL;DR

Step I : Se faire passer pour un développeur Amazon

`Kindle4NTHacking.tar.gz` Fake Dev Key + Diagnostic mode + Script ✨

`CTRL + C` `CTRL + V` via USB

Jailbreak TL;DR

Step I : Se faire passer pour un développeur Amazon

`Kindle4NTHacking.tar.gz` Fake Dev Key + Diagnostic mode + Script ✨

`CTRL + C` `CTRL + V` via USB

On Kindle `Menu → Settings → Menu → Restart`

Jailbreak TL;DR

Step I : Se faire passer pour un développeur Amazon

`Kindle4NTHacking.tar.gz` Fake Dev Key + Diagnostic mode

`CTRL + C` `CTRL + V` via USB

On Kindle `Menu → Settings → Menu → Restart`

Jailbreak TL;DR

- [x] Step I : Se faire passer pour un développeur Amazon

Step II : Connexion SSH

Jailbreak TL;DR

- [x] Step I : Se faire passer pour un développeur Amazon

Step II : Connexion SSH

- Package USBNetwork `kindle-usbnetwork-0.57.N-k4.zip`

Jailbreak TL;DR

☑ Step I : Se faire passer pour un développeur Amazon

Step II : Connexion SSH

Package USBNetwork `kindle-usbnetwork-0.57.N-k4.zip`

```
#!/bin/sh
# NOTE: The K4 has a specific default
HOST_IP=192.168.15.201
KINDLE_IP=192.168.15.244
# Allow SSH over WiFi
# NOTE: If you set this to true, the SSHD *WILL* check your passwords!
K3_WIFI="false"
K3_WIFI_SSHD_ONLY="false"
```

Jailbreak TL;DR

- [x] Step I : Se faire passer pour un développeur Amazon
- [x] Step II : Connexion SSH
- [ ] Step III : Virer Amazon de mon Kindle

Jailbreak TL;DR

- [x] Step I : Se faire passer pour un développeur Amazon
- [x] Step II : Connexion SSH
- [ ] Step III : Virer Amazon de mon Kindle

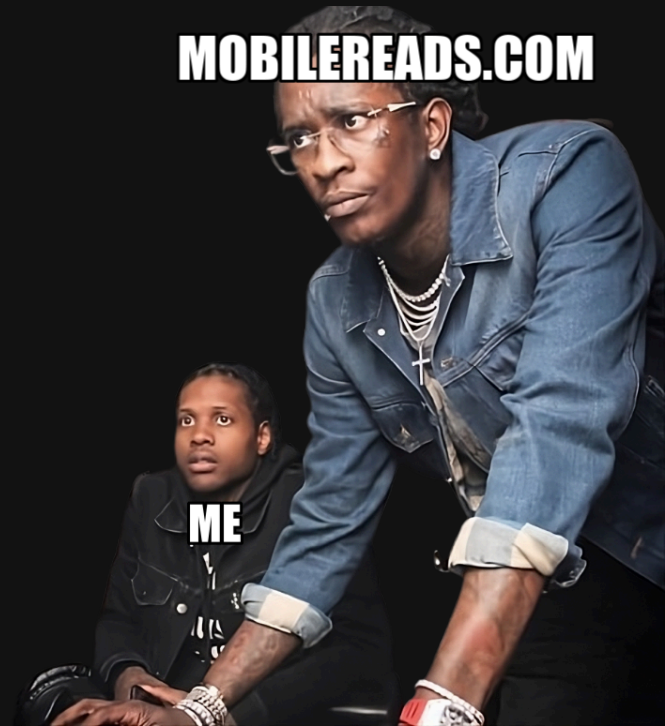
    No Internet updates: `/etc/uks` into `/etc/uks.disabled`

Jailbreak TL;DR

- [x] Step I : Se faire passer pour un développeur Amazon
- [x] Step II : Connexion SSH
- [x] Step III : Virer Amazon de mon Kindle

Jailbreak TL;DR

- [x] Step I : Se faire passer pour un développeur Amazon
- [x] Step II : Connexion SSH
- [x] Step III : Virer Amazon de mon Kindle



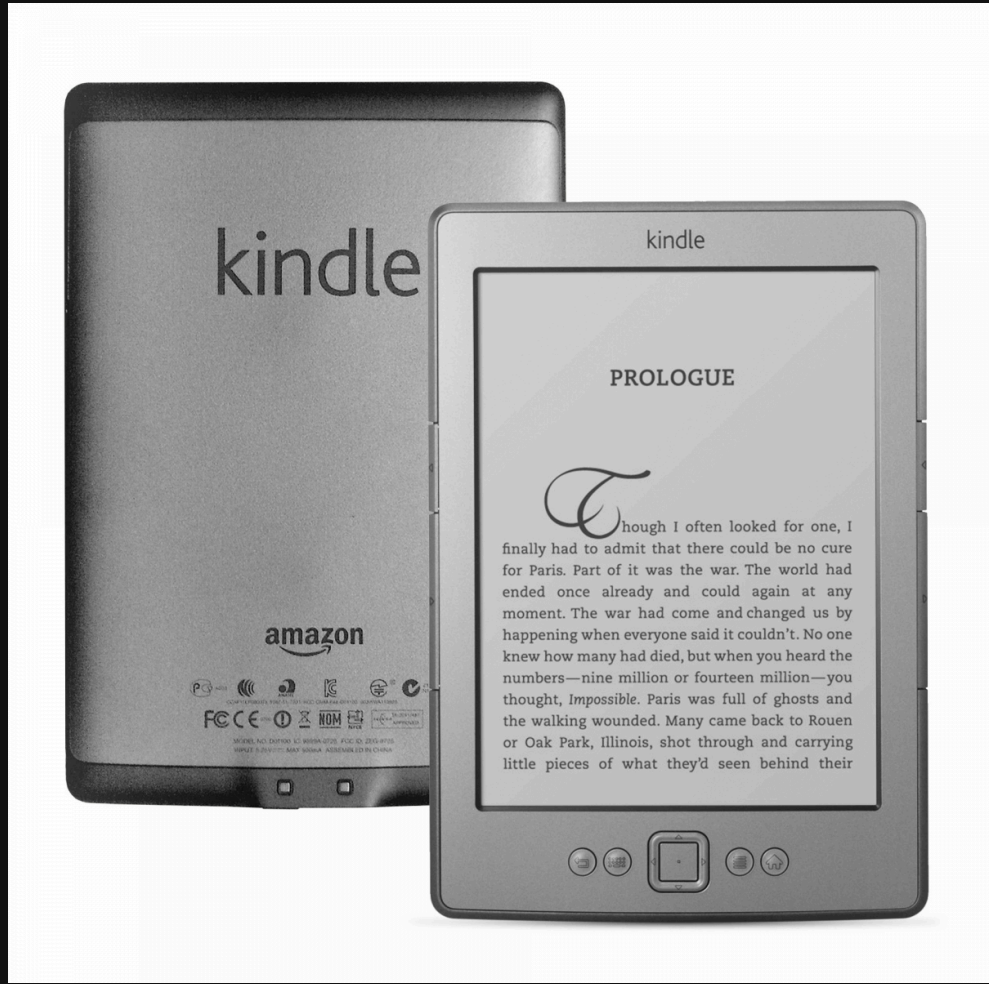
```
touch ./hello.c
```

```
#include <stdio.h>

int main() {
   printf("Yo la TZ");
   return 0;
}
```

```
gcc -o hello mysourcefile.c
```

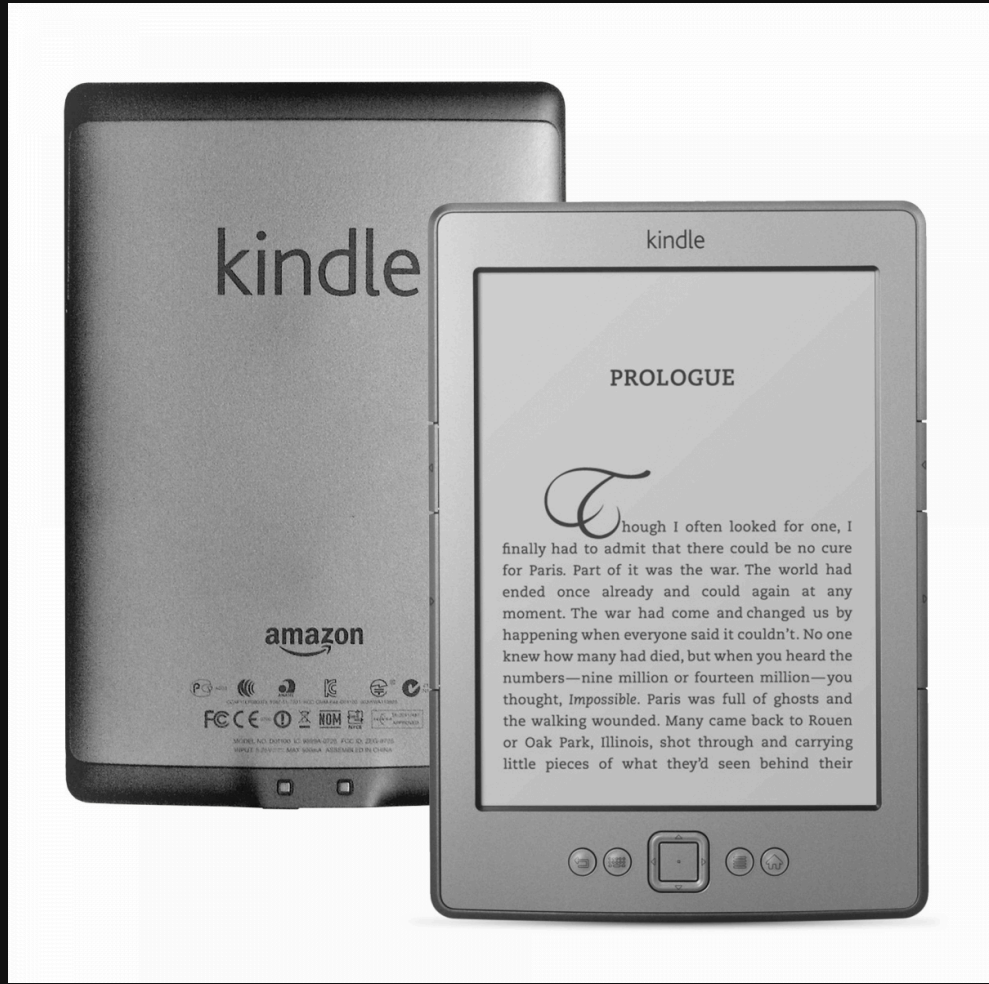
Kindle 4th Gen

Release	`September 2011`
Display	`6"` `600×800p` `8 bit Greyscale`
OS	`Linux (Amazon)`
Memory	`256 MB DDR`
CPU	`ARM Cortex-A8` `800MHz` `Freescale i.MX508`



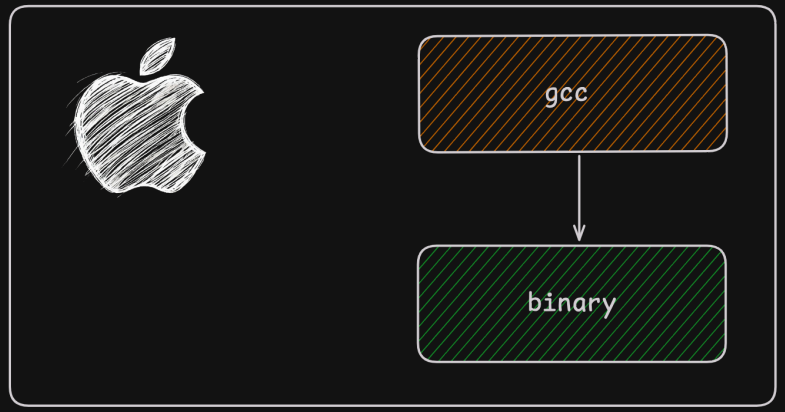
Kindle 4th Gen

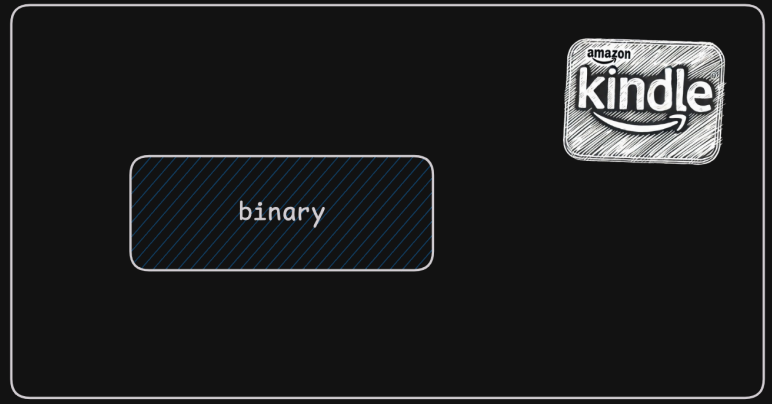
Release	`September 2011`
Display	`6"` `600×800p` `8 bit Greyscale`
OS	`Linux (Amazon)`
Memory	`256 MB DDR`
CPU	`ARM Cortex-A8` `800MHz` `Freescale i.MX508`

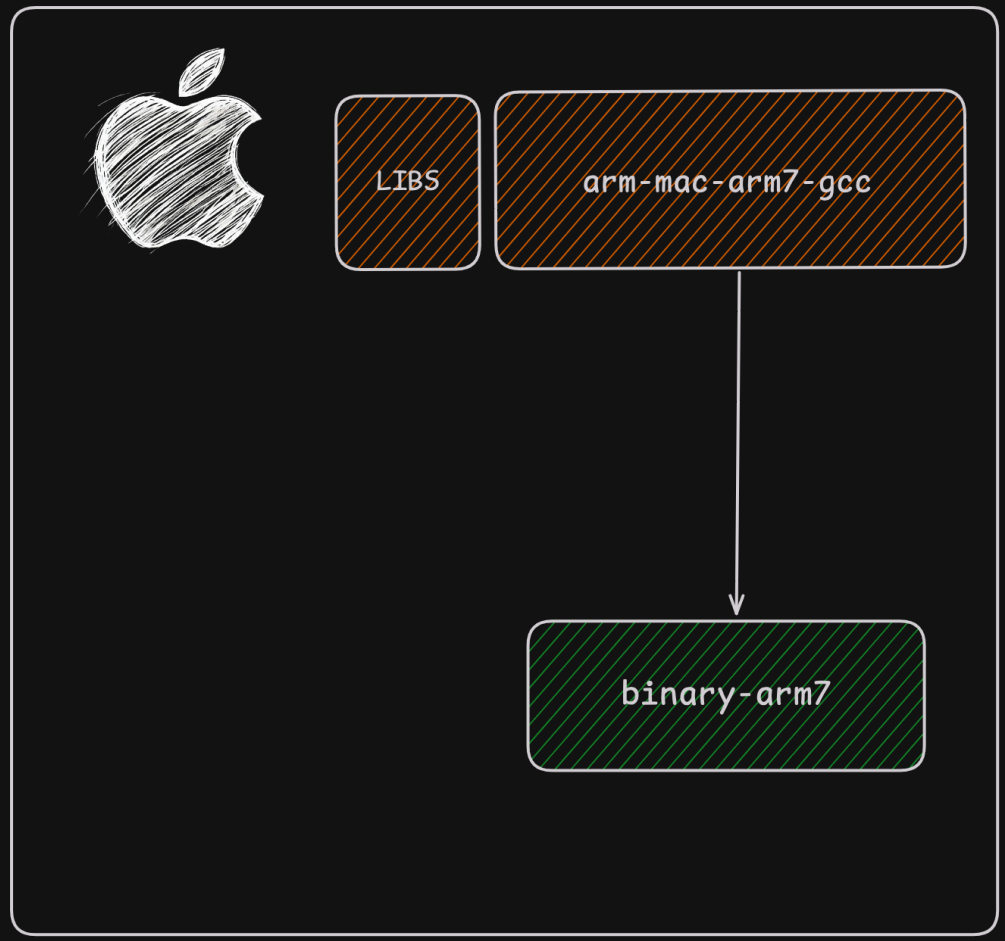


CROSS COMPILATION

```
ARCH=$(uname -m); 0x01000011 TGT=${ARCH/-/_}
dd if=/dev/urandom bs=512 count=2048 2>/dev/

if [ "$BYTE_ORDER" == "LE" ]; then
    dd if="$BIN" bs=0x05000001 skip=1 count=
else
    echo "Big endian detected, no swap requi
fi

<<EOF sed 's|$|&|' | tr -d '\n' | dd bs=1 of
Blob $(wc -c <$F/blob)
EOF

echo '#include<stdio.h>' > $F/x.c
echo 'int main(){return 0;}' >> $F/x.c

bytecode: [ 0x01000064, 0x02000002,0xFF000000
  uint32_t x = 0x12345678;
```


gcc
binary


amazon
kindle
binary


LIBS
arm-mac-arm7-gcc
binary-arm7

Crosstool-NG can build all these kinds of toolchains, or is aiming at it, anyway

- CROSSTOOL-NG OFFICIAL DOC

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

Step I : Installer crosstool-NG sur macOS

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

Step I : Installer crosstool-NG sur macOS

`brew install` beaucoup de trop de packages

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

Step I : Installer crosstool-NG sur macOS

`brew install` beaucoup de trop de packages

Monter un disque virtuel entier sensible a la casse pour crosstool-NG 👀

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

Step I : Installer crosstool-NG sur macOS

`brew install` beaucoup de trop de packages

Monter un disque virtuel entier sensible a la casse pour crosstool-NG 👀

J'ai failli briquer mon Mac

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

- [x] ~~Step 1 : Installer crosstool-NG sur macOS~~

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

~~Step I : Installer crosstool-NG sur macOS~~

Step I : Installer crosstool-NG dans Docker

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

~~Step I : Installer crosstool-NG sur macOS~~

Step I : Installer crosstool-NG dans Docker

Installer crosstool-NG sur Ubuntu

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

~~Step I : Installer crosstool-NG sur macOS~~

Step I : Installer crosstool-NG dans Docker

Installer crosstool-NG sur Ubuntu

Configurer la toolchain arm-cortex_a8-linux-gnueabi-gcc 🛠️

Architecture	`Armv7`	C lib	`glibc`
Compiler	`gcc`	CC version	`linaro 4.17`
Interface	`EABI`	Endianness	`Little endian`

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

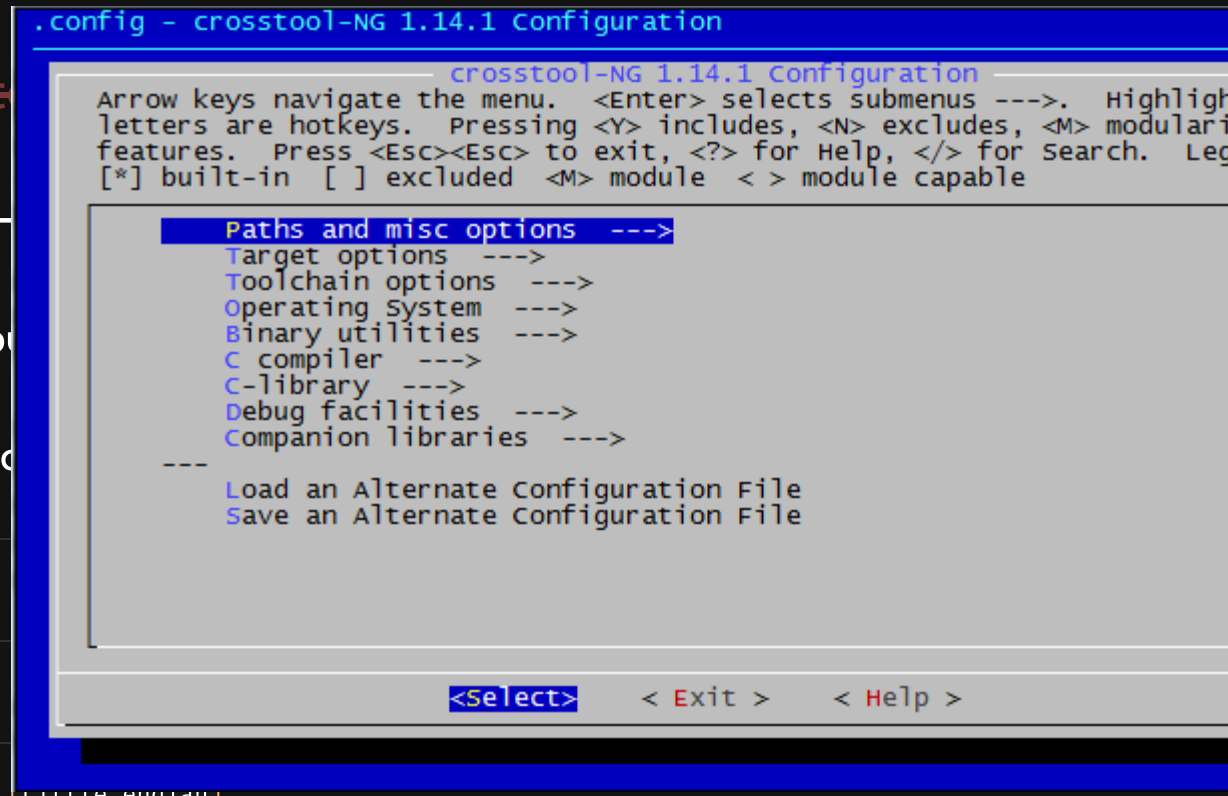
~~Step I : Installer crosst~~

Step I : Installer crosstool-

Installer crosstool-NG sur Ubu

Configurer la toolchain arm-c

Architecture	`Armv7`	C lib	
Compiler	`gcc`	CC version	
Interface	`EABI`	Endianness	`Little endian`

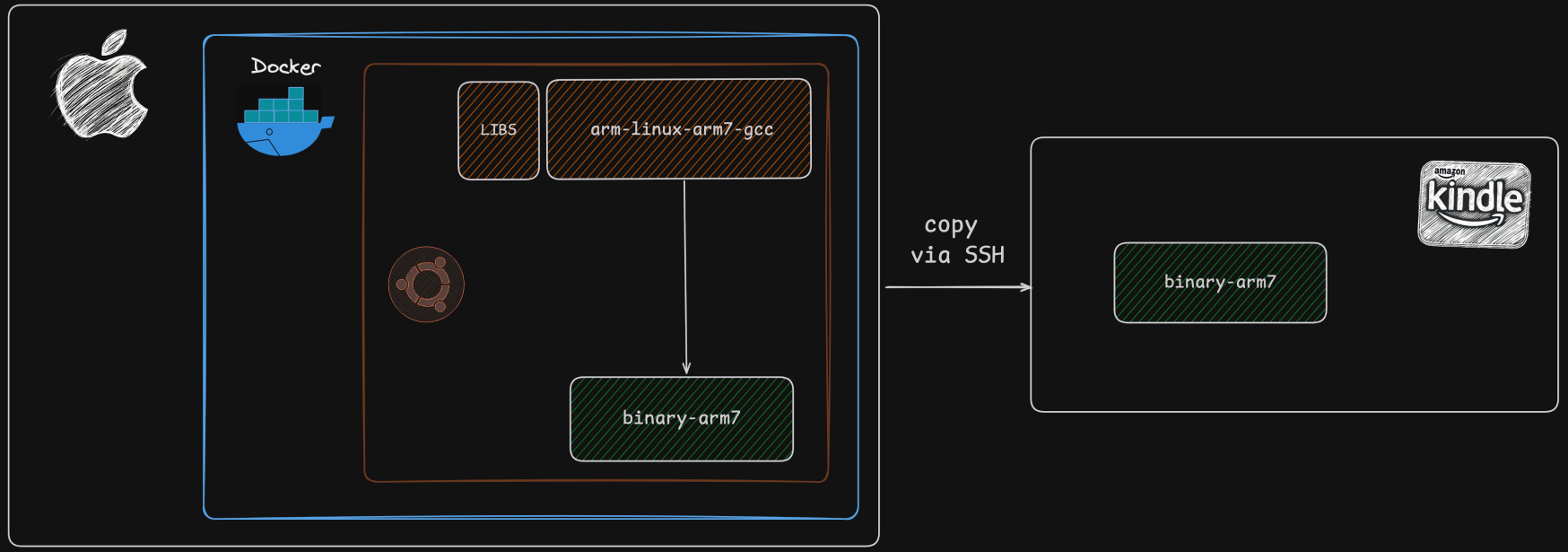


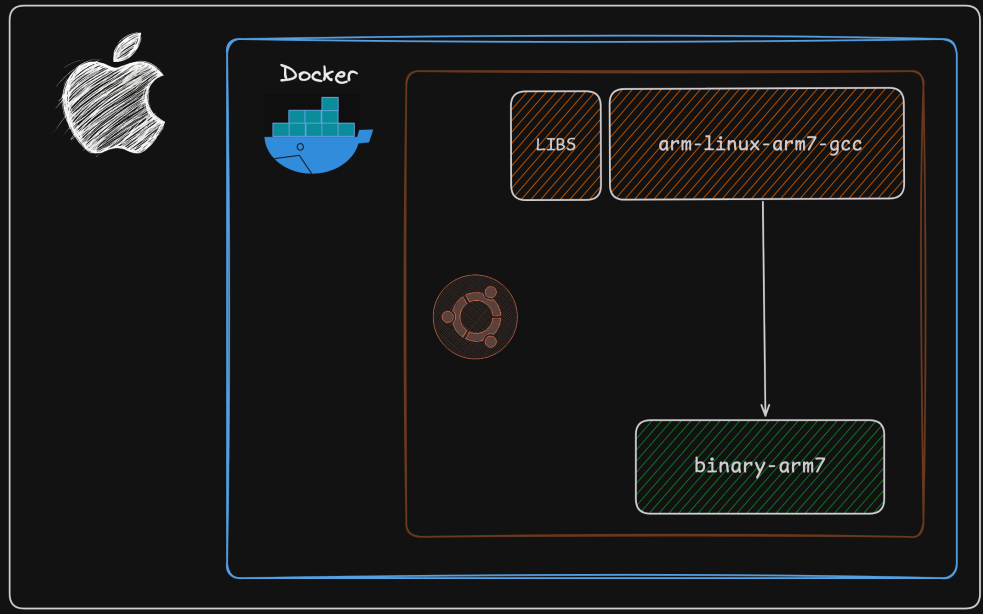
Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

- [x] ~~Step I : Installer crosstool-NG sur macOS~~
- [x] Step I : Installer crosstool-NG dans Docker

Toolchain `arm-cortex_a8-linux-gnueabi-gcc`

- [x] ~~Step I : Installer crosstool-NG sur macOS~~
- [x] Step I : Installer crosstool-NG dans Docker
- [ ] Step II : Générer la toolchain


Docker
LIBS
arm-linux-arm7-gcc
binary-arm7
copy
via SSH
amazon
kindle
binary-arm7


Docker
LIBS
arm-linux-arm7-gcc
binary-arm7

copy
via SSH


amazon
kindle
binary-arm7


OVER ENGINE

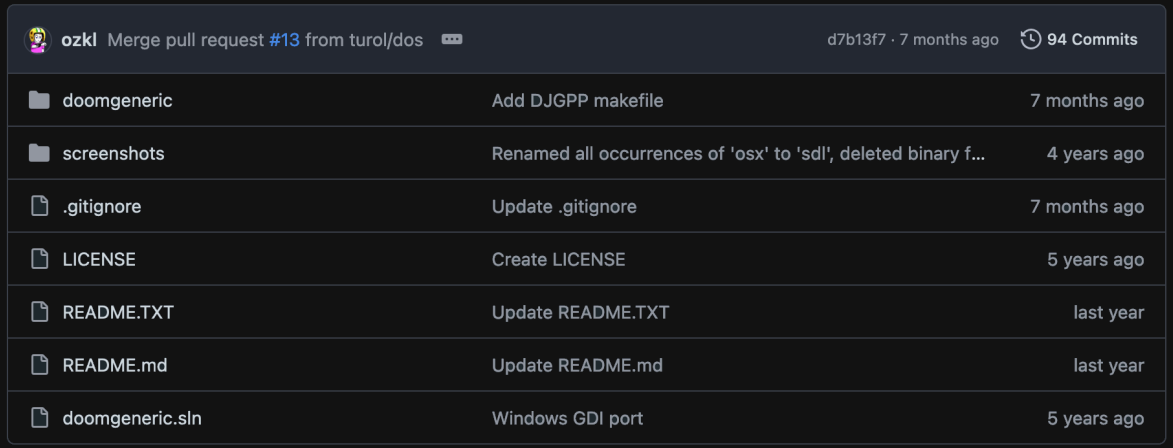
GAME ENGINE

- Rendering
- Input/Output
- Game Logic

.WAD

- Textures
- Level Maps
- Metadata

ozkl/doomgeneric



ozkl Merge pull request #13 from turol/dos — d7b13f7 · 7 months ago — 94 Commits

doomgeneric	Add DJGPP makefile	7 months ago
screenshots	Renamed all occurrences of 'osx' to 'sdl', deleted binary f...	4 years ago
.gitignore	Update .gitignore	7 months ago
LICENSE	Create LICENSE	5 years ago
README.TXT	Update README.TXT	last year
README.md	Update README.md	last year
doomgeneric.sln	Windows GDI port	5 years ago

README | GPL-2.0 license

doomgeneric

The purpose of doomgeneric is to make porting Doom easier. Of course Doom is already portable but with doomgeneric it is possible with just a few functions.

To try it you will need a WAD file (game data). If you don't own the game, shareware version is freely available (doom1.wad).

About

Easily portable doom

game doom

Readme
GPL-2.0 license
Activity
1.3k stars
16 watching
157 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

C 99.3% Other 0.7%

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

doomgeneric_kindle.c

```c
// Initialize your platfrom (create window, framebuffer, etc ... )
void DG_Init() {}

// This is for setting the window title as Doom sets this from WAD file.
void DG_SetWindowTitle() {}

// Provide keyboard events
void DG_GetKey() {}

// Sleep in milliseconds.
void DG_SleepMs() {}

// Frame is ready in DG_ScreenBuffer. Copy it to your platform's screen.
void DG_DrawFrame() {}
```

Est-ce que c'est de la triche ?

Peut-être, mais je fais ce que je veux

Peut-être, mais je fais ce que je veux

- MOI

Framebuffer `/dev/fb0`

Framebuffer `/dev/fb0`

Framebuffer `/dev/fb0`

Chaque pixel:

Framebuffer `/dev/fb0`

Chaque pixel:

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded
- ~~*50 Shades Of Grey*~~
- 256 nuances de gris possibles

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded
- ~~*50 Shades Of Grey*~~
- 256 nuances de gris possibles

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded
- ~~*50 Shades Of Grey*~~
- 256 nuances de gris possibles

128

10000000

Framebuffer `/dev/fb0`

Chaque pixel:

- 8 bit encoded
- ~~*50 Shades Of Grey*~~
- 256 nuances de gris possibles

- Raw binary into `/dev/fb0`

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

```c
void DG_Init() { frameBuffer = open("/dev/fb0", O_RDWR); }

void DG_DrawFrame() {
    uint8_t memory_framebuffer[SCREEN_WIDTH * SCREEN_HEIGHT] = {0};

    for (int y = 0; y < DOOMGENERIC_RESY && y < SCREEN_HEIGHT; y++) {
        for (int x = 0; x < DOOMGENERIC_RESX && x < SCREEN_WIDTH; x++) {
            pixel_t pixel = DG_ScreenBuffer[y * DOOMGENERIC_RESX + x];

            // RGBA bits
            uint8_t red = (pixel >> 16) & 0xFF;
            uint8_t green = (pixel >> 8) & 0xFF;
            uint8_t blue = pixel & 0xFF;
            uint8_t intensity = (0.299 * red) + (0.587 * green) + (0.114 * blue);

            memory_framebuffer[y * SCREEN_WIDTH + x] = 255 - intensity;
        }
    }

    // Write to fb0
    write(frameBuffer, memory_framebuffer, sizeof(memory_framebuffer));

    // update display with eink_fb
    FILE *file = fopen("/proc/eink_fb/update_display", "w");
    fprintf(file, "1");
    fclose(file);
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0×FF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0xFF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0×FF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0xFF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0xFF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0×FF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0×FF;
    return 1;
}
```

Inputs `/dev/input/event1`

```
void pollInputs() {
    struct input_event ev;
    while (read(input_fd, &ev, sizeof(ev)) > 0) {
        unsigned char doomKey;
        switch (ev.code) {
            case KEY_LEFT: doomKey = KEY_LEFTARROW; break;
            case KEY_RIGHT: doomKey = KEY_RIGHTARROW;break;
            case KEY_UP: doomKey = KEY_UPARROW;break;
            case KEY_DOWN: doomKey = KEY_DOWNARROW;break;
            case KEY_ENTER: doomKey = KEY_FIRE;break;
            default: continue;
        }
         keyQueue[writeIndex] = doomKey;
         writeIndex++;
    }
}

int DG_GetKey(unsigned char* doomKey) {
    unsigned short keyData = keyQueue[readIndex];
    readIndex++;
    *doomKey = keyData & 0×FF;
    return 1;
}
```

vitesse write framebuffer >
vitesse maj e-ink

Display mode rates

Display mode rates

Greyscale 8 bit mode **~600ms**

$$I_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

Display mode rates

Greyscale 8 bit mode **~600ms**

$$I_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

B/W 1 bit mode **~125ms**

Display mode rates

Greyscale 8 bit mode **~600ms**

$$I_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

B/W 1 bit mode **~125ms**

$$I_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

Display mode rates

Greyscale 8 bit mode **~600ms**

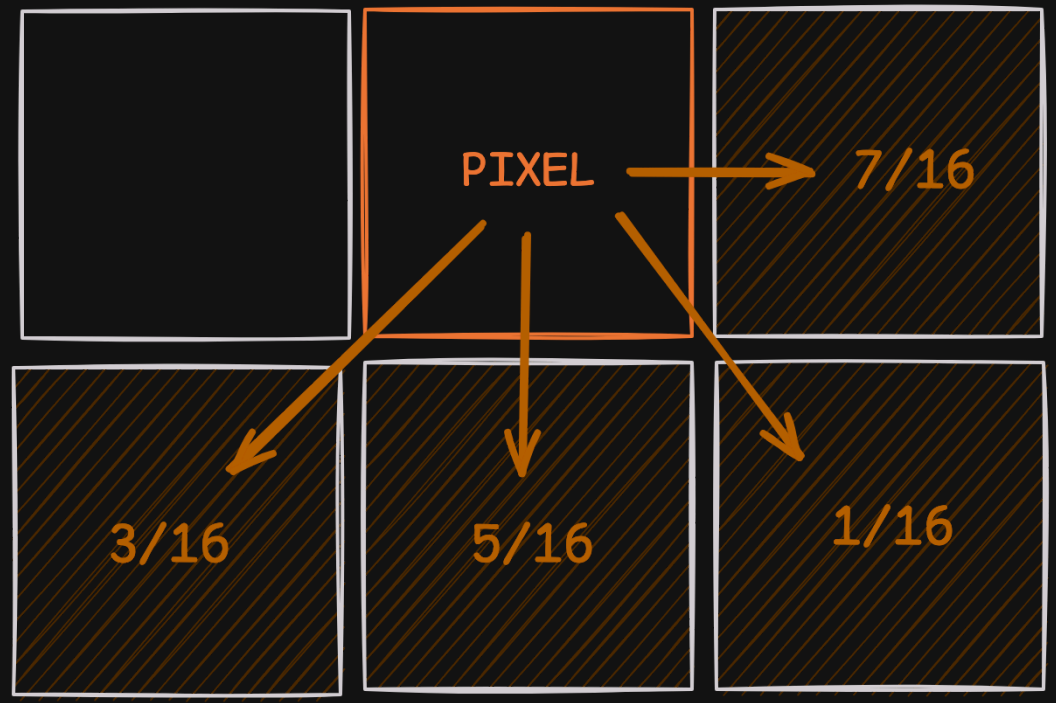
$$I_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

B/W 1 bit mode **~125ms**

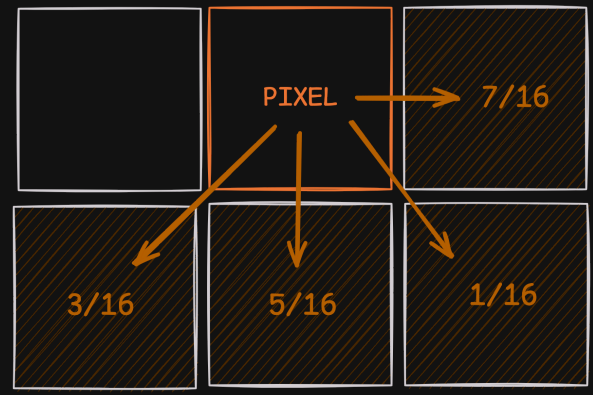
$$I_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

$$C_{\text{Black/White}} = 0 \longleftarrow I_{\text{gray}} \longrightarrow 255$$

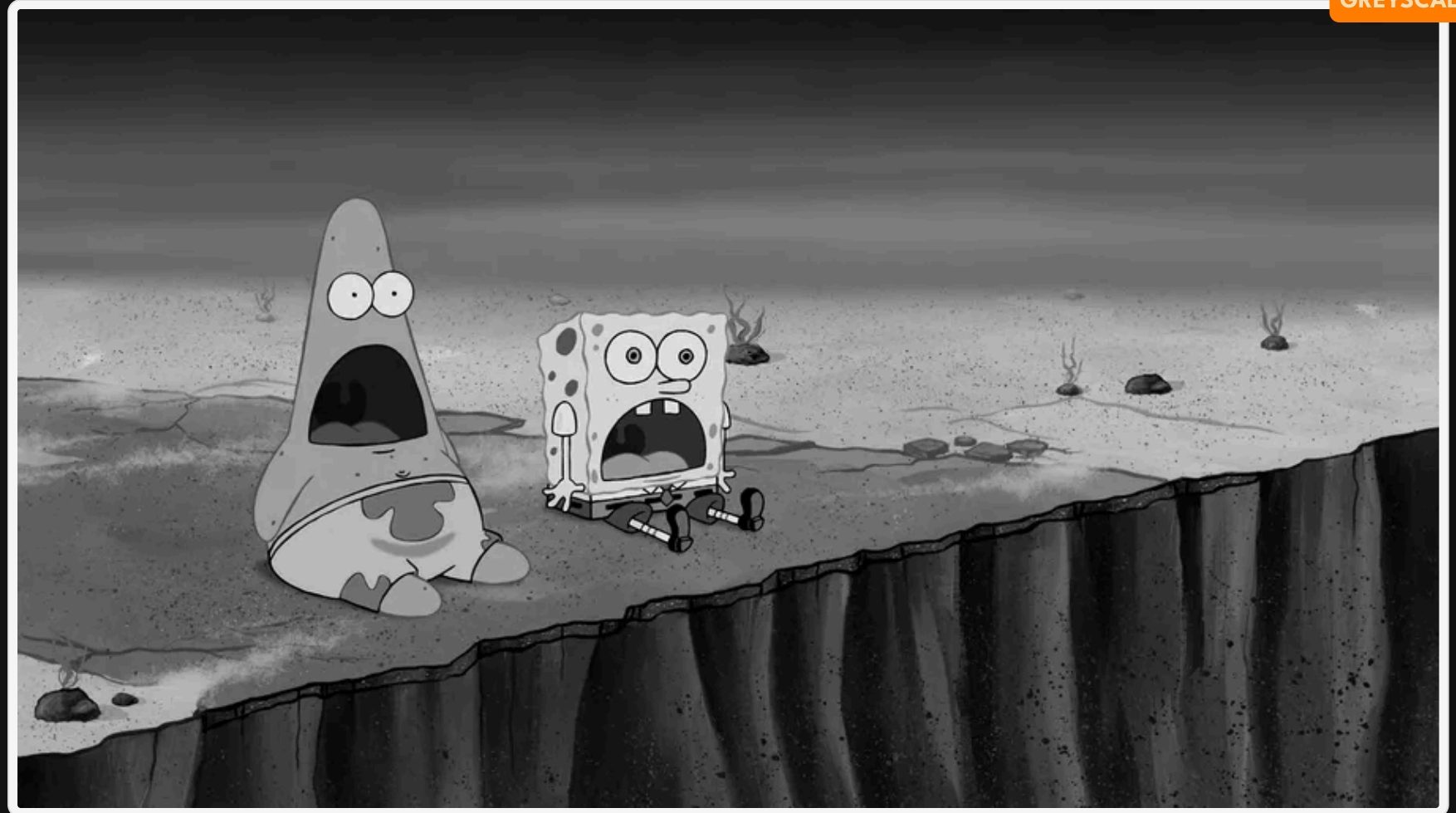
Dithering Error Diffusion

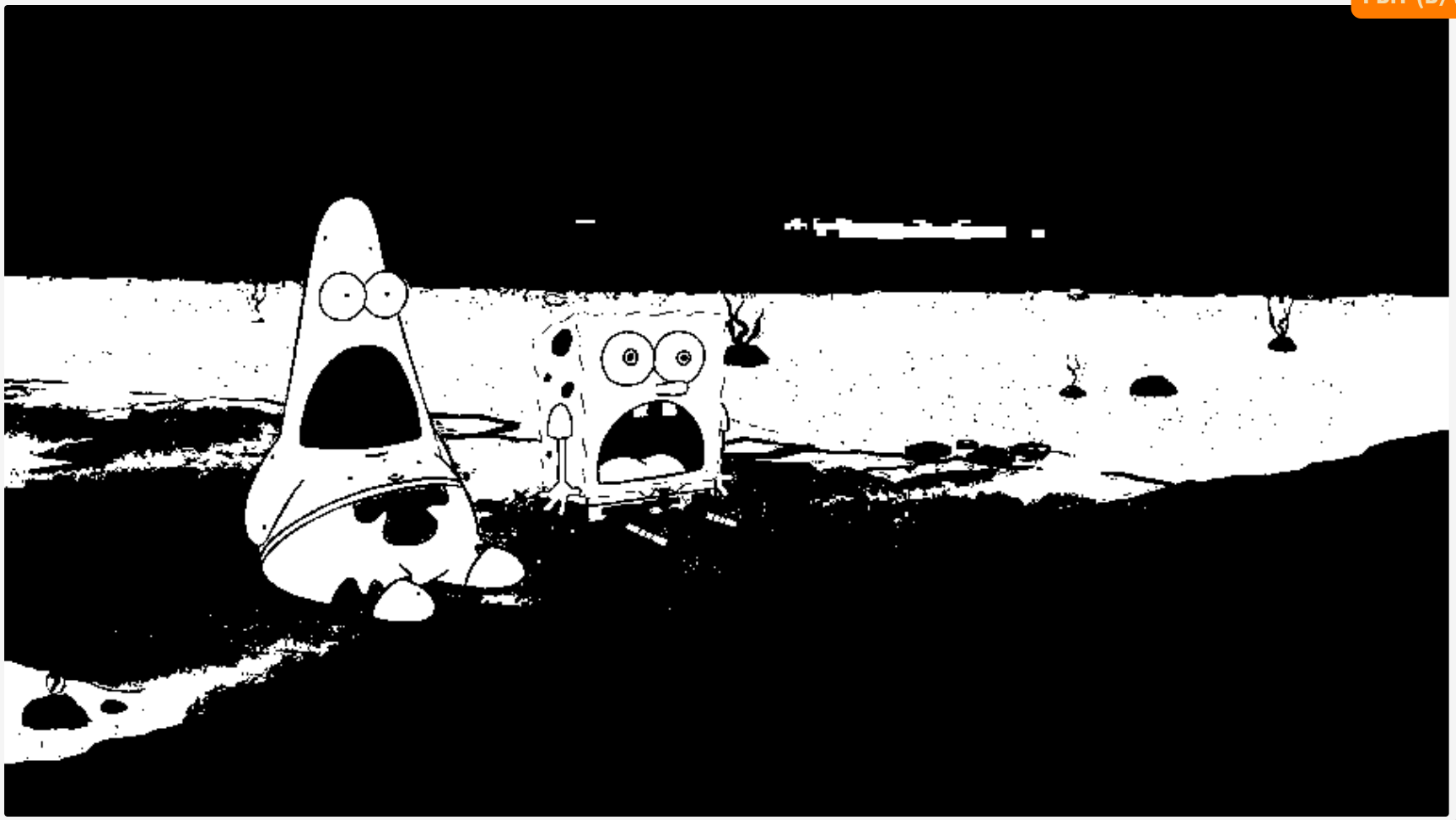


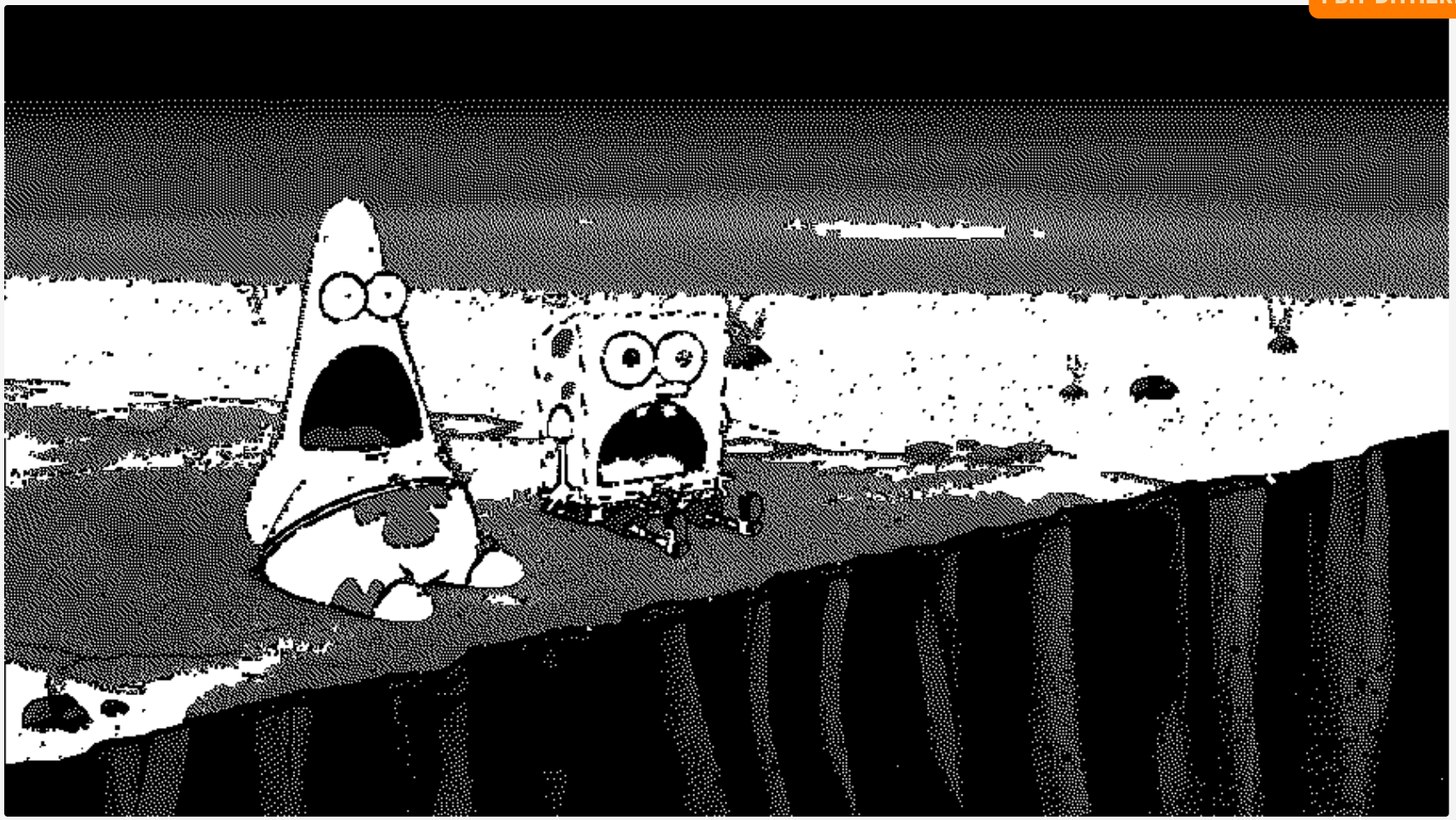
50	100	190	40	240
200	120	5	200	10
50	170	35	100	125
250	65	130	170	200
150	55	220	40	120




ORIGINAL


GREYSCALE


1 BIT (B/W)


1 BIT DITHERED

What I learned ? / Victory animation

Moustapha AGACK

Full Stack Developer (and lots of Cloud)

Smash my keyboard for a living—results vary, but sometimes cool stuff happens

@Zenika Lyon + @Homeserve

bento.me/mouss

Twitter @mousstoh

moussto/talk-k4-doom-slides

moussto/k4-doom

S/O ♡

Game Engine Black Book DOOM	Fabien Sanglard	
Kindle Developer's Corner	Mobilereads.com	
Doomgeneric	ozkl/doomgeneric	
Toolchain Types	Crosstool-NG	
Kindle Hacks	Sotiris Papatheodorou	
Dithering Eleven Algorithms	Tanner Helland	
Error Diffusion Dithering	Computerphile	
Dithering for Eink Display Panel	Daiyu Ko	
Zenika Friends 🙏🏾		