Old Game tech

Old Game tech i

Golden Sun (2001)





Figure 2: Golden Sun Battle

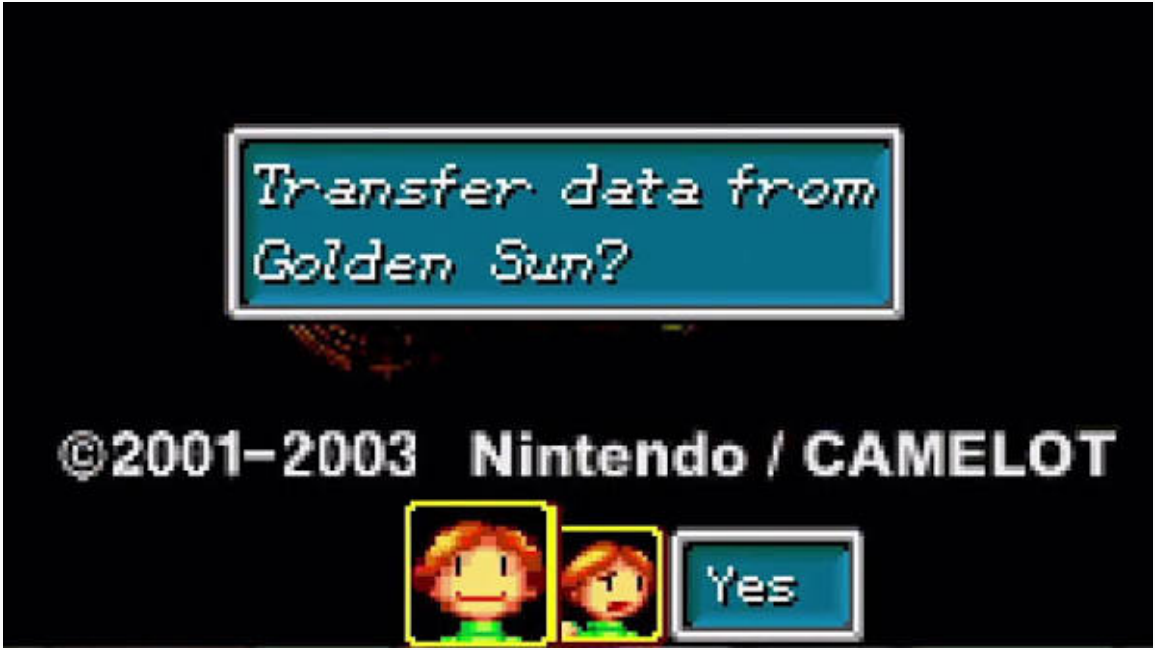


Figure 3: Golden sun save screen

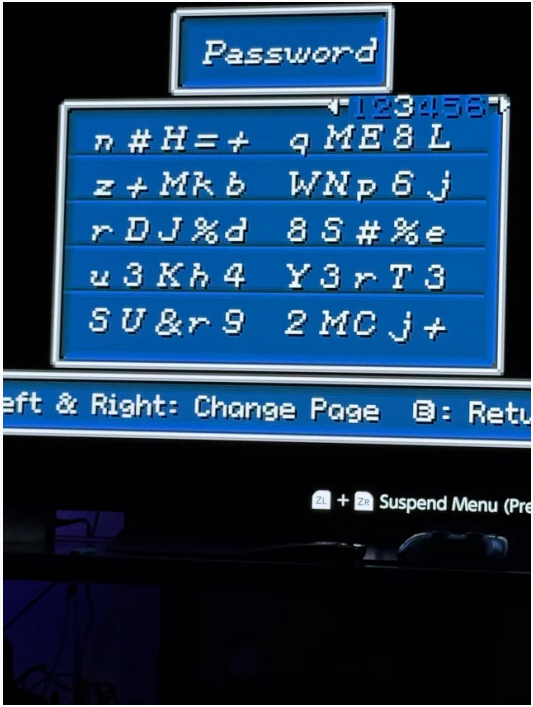


Figure 4: Golden Sun save screen

How much to transfer

You could copy over your characters and items from *Golden Sun* to *The Lost Age* with a password. There are 3 tiers:

Tier	Length	What moves
Bronze	16 chars	Levels, Djinn, event flags, utility Psynergy items
Silver	61 chars	Bronze data + character stats
Gold	260 chars	Silver data + inventories and coins

How to:

1. Encode bytes as 6-bit values, most-significant bit first, using `ABCDEFGHJKLMNPQRSTUVWXYZ23456789abcdefghijkmnpqrstuvwxyz!?#&$%+=`.
2. After each 9 values, add a check value: the sum of those nine % 64. This catches off by one errors.
3. Offset each value by its position modulo 64, then map the result through the alphabet. The final byte is an XOR key applied to all earlier bytes before conversion; decoding reverses both.
4. Verify the crc16 stored before the XOR step.

The per group checks locate typos, and the CRC validates the payload.

Codec details: Sol Cipher Make your own password

Byte codec

```python
ALPHABET = "ABCDEFGHJKLMNPQRSTUVWXYZ23456789abcdefghijkmnpqrstuvwxyz!?#&$%+="
SIZES = {16: 11, 61: 41, 260: 175}  # password chars -> bytes including CRC

def crc16(data):
    crc = 0xFFFF
    for byte in data:
        crc ^= byte << 8
        for _ in range(8):
            crc = (((crc << 1) - 0x1021) if crc & 0x8000 else (crc << 1)) & 0xFFFF
    return (~crc) & 0xFFFF

def encode(payload):
    crc = crc16(payload)
    raw = bytearray(payload + crc.to_bytes(2, "big"))
    key = raw[-1]
    raw[:-1] = bytes(b ^ key for b in raw[:-1])

    bits = "".join(f"{b:08b}" for b in raw)
    values = [int(bits[i:i + 6].ljust(6, "0"), 2) for i in range(0, len(bits), 6)]
```

Pokemon Gold/Silver (2000)

Pokemon Gold/Silver (2000)





Figure 6: English Pokémon Gold clock setup screen: "What time is it?"

(Setting the time at the start of the game)



Figure 7: Pokémon Gold and Silver clock prompts with the same scene at night and in daylight

The clock in the cartridge

The Game Boy itself had no clock, so *Pokémon Gold* and *Silver* came with a clock. It kept running while the handheld was off even. Also it skewed quite a lot time wise since it wasn't synced with NTP.

Some Pokémon appeared at particular times, events depended on the day, and friendship evolutions could differ between day and night. Also your mom annoyed you at certain times and wasted your money.



Figure 8: Inside a Pokémon Silver cartridge, showing the MBC3 chip and CR2025 battery

Pokemon stadium (1999)

Pokemon stadium (1999)



Figure 9: Drowzee kick



Figure 10: Back of the Nintendo 64 Transfer Pak, showing the Game Boy cartridge slot

Pick a mode


ASH
ID 06020
ASH
ID 44669
RED
ID 20152
1P
2P
3P
Please select a Game Boy Game Pak.

Emulation


STADIUM
STADIUM
The STADIUM where tournaments are held. Test the mettle of your Pokémon with four sets of rules.

Bringing the GBC to Rust

There are three levels of writing your own GBC games:

- ASM: RGBDS
- C: GBDK
- Rust: rust-gb
  - fork of rust (with "z80") support
  - fork of llvm (with "z80") support
  - links to emit a gb cartridge

The author of rust-gb has a fork of cranelift, where they wanted to use cranelift:

cranelift-z80

Why not make my own? (really difficult)

Building my own

Project: sangoku

- Has a small fork (50 loc) of rust
- fork of rustc's backend that emits cranelift IR
- fork of cranelift to optimize instructions for an 8-bit backend
- Tedious optimization: `core::fmt` made the binary $>$ 32kb
- Had to remove panics with `panic = "immediate-abort"`, otherwise $>$ 32kb
- Functions that allocate, like `sort` would emit 90kb of alloc/fmt code
- Any unicode lookup functions (e.g to_lower) would use a lookup table and be 60kb
- no multiply or divide, so you have to emulate in software

Audio

- pret's original asm
- Surfing Source
- Surfing
- Emulator

The Game boy has 4 channels, so you can roughly play 4 notes at a time

- CH1 + CH2: Pulse (fixed)
- CH3: Wave, user supplied waves
- CH4: Noise, random waves

All the rest: TBD

- Reading through Game boy coding adventure