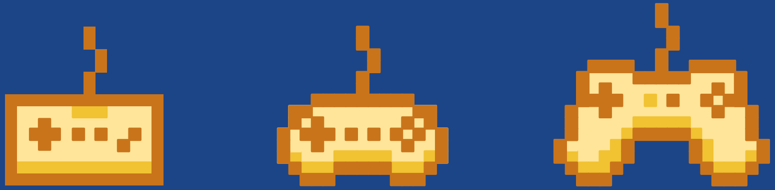


ReAnimate

From Code Lines to Creative Leaps: The Evolution of Game Development

By Carlos Pinto Gomez

CONTEXT

What we won't cover

Deep dive into certain technologies
Deep history facts
AA/AAA game creation

What I hope to convey

Limitations of the eras
Game development pivoting moments
Evolution of game creation tools

TABLE OF CONTENTS

TIMELINE

1970s … Today

CASE STUDIES

Key games and their reality

TOOLS

Programming Languages, Engines

PRODUCTION

Teams, time, work style

INDUSTRY

Risks, reality, trends

1970s

BARE METAL PROGRAMMING

1950s-1970s

1958

Tennis for Two

Spacewar

1962

1972

Magnavox Odyssey

Space Invaders

1978

PROJECT

Tomohiro Nishikado, creator of Space Invaders

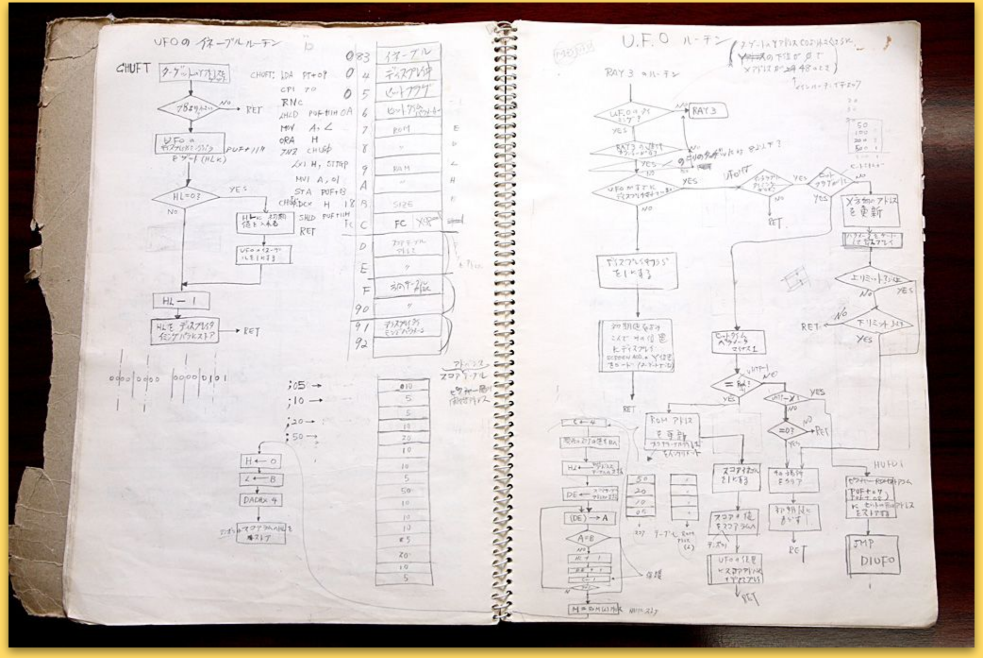
Nishikado worked on all parts of the game: code, hardware, sound, etc.

It took about 1 year to create through his own iterations.

Deployable on ROM, which means patching is done directly on hardware.

Shipped through physical arcades.

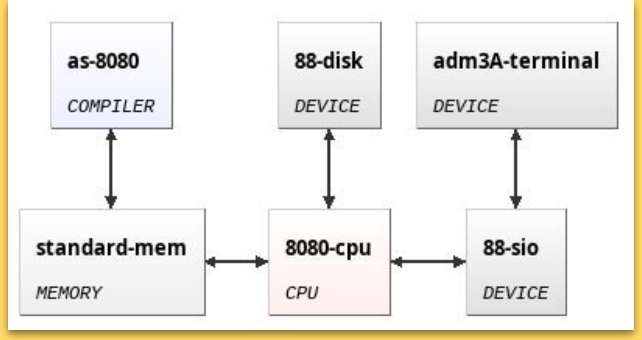
SAMPLE ROUTINE DESIGN



Tomohiro Nishikado's sample subroutine flow diagram

MITS ALTAIR 8800

MITS Altair 8800



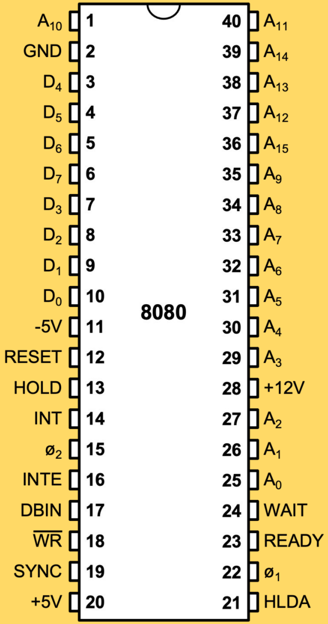
System architecture of the Altair 8800

Personal computers were more powerful than consoles.

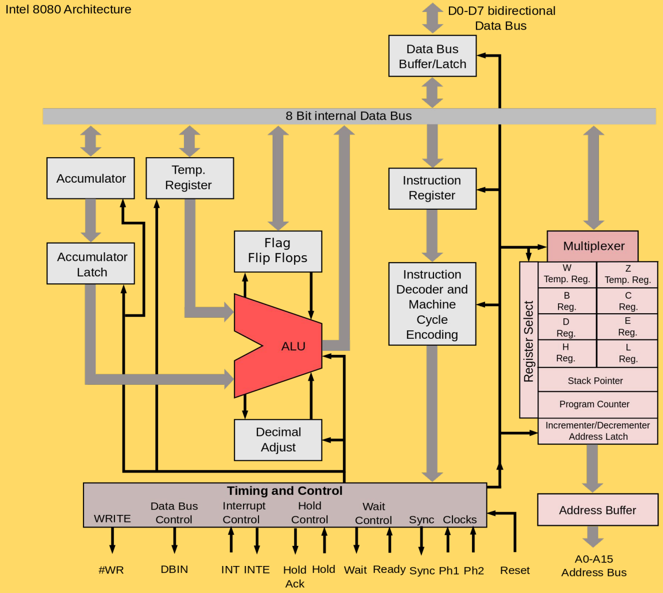
Specifications:

- Processor: Intel 8080 @2MHz
- RAM: from 256 bytes to 64 kB
- Storage (optional): paper tapes, cassette tapes, floppy disks
- Video memory: none
- Resolution: 256x224

ARCHITECTURE

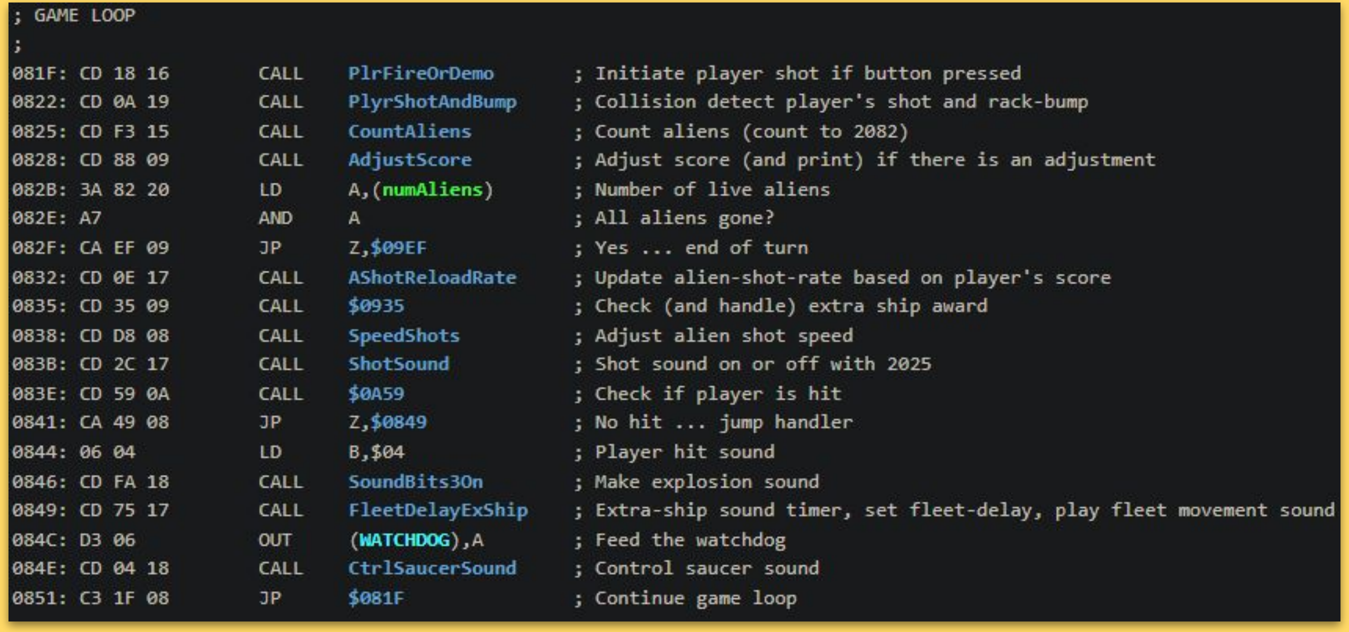


Intel 8080 pins



Intel 8080 Architecture

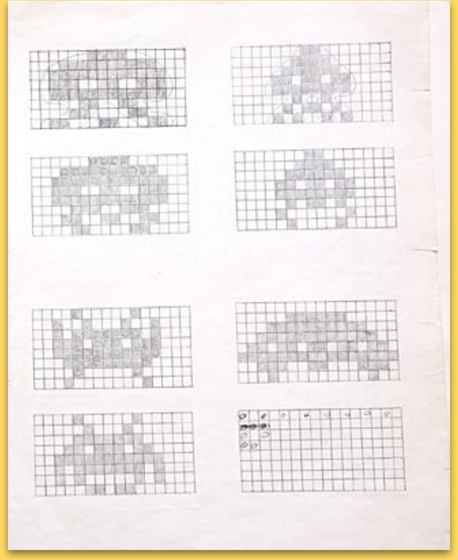
GAME LOOP



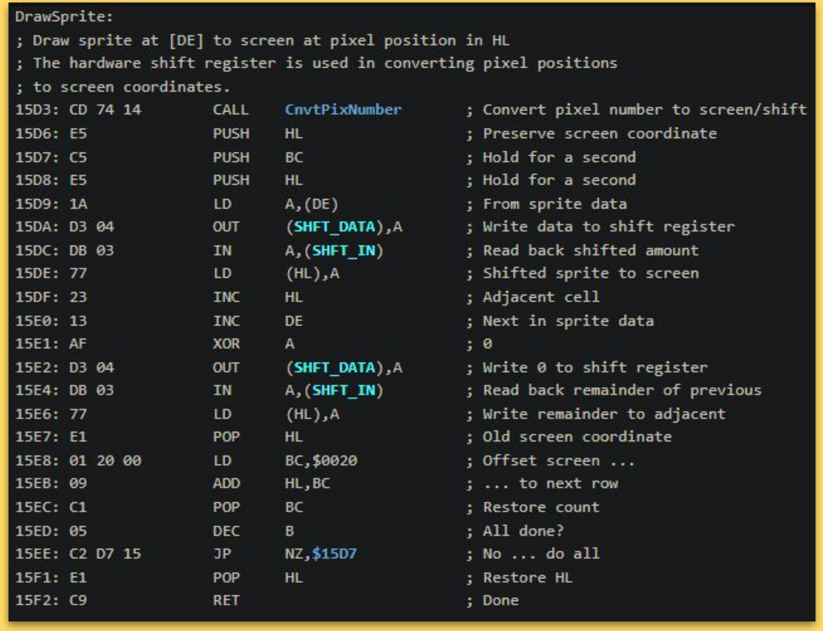
```
; GAME LOOP
;
081F: CD 18 16       CALL    PlrFireOrDemo       ; Initiate player shot if button pressed
0822: CD 0A 19       CALL    PlyrShotAndBump     ; Collision detect player's shot and rack-bump
0825: CD F3 15       CALL    CountAliens         ; Count aliens (count to 2082)
0828: CD 88 09       CALL    AdjustScore         ; Adjust score (and print) if there is an adjustment
082B: 3A 82 20       LD      A,(numAliens)       ; Number of live aliens
082E: A7             AND     A                   ; All aliens gone?
082F: CA EF 09       JP      Z,$09EF             ; Yes ... end of turn
0832: CD 0E 17       CALL    AShotReloadRate     ; Update alien-shot-rate based on player's score
0835: CD 35 09       CALL    $0935               ; Check (and handle) extra ship award
0838: CD D8 08       CALL    SpeedShots          ; Adjust alien shot speed
083B: CD 2C 17       CALL    ShotSound           ; Shot sound on or off with 2025
083E: CD 59 0A       CALL    $0A59               ; Check if player is hit
0841: CA 49 08       JP      Z,$0849             ; No hit ... jump handler
0844: 06 04          LD      B,$04               ; Player hit sound
0846: CD FA 18       CALL    SoundBits3On        ; Make explosion sound
0849: CD 75 17       CALL    FleetDelayExShip    ; Extra-ship sound timer, set fleet-delay, play fleet movement sound
084C: D3 06          OUT     (WATCHDOG),A        ; Feed the watchdog
084E: CD 04 18       CALL    CtrlSaucerSound     ; Control saucer sound
0851: C3 1F 08       JP      $081F               ; Continue game loop
```

Zilog Z80 assembly language

ART



Space Invaders pixel drawing



```
DrawSprite:
; Draw sprite at [DE] to screen at pixel position in HL
; The hardware shift register is used in converting pixel positions
; to screen coordinates.
15D3: CD 74 14        CALL    CnvtPixNumber      ; Convert pixel number to screen/shift
15D6: E5              PUSH    HL                 ; Preserve screen coordinate
15D7: C5              PUSH    BC                 ; Hold for a second
15D8: E5              PUSH    HL                 ; Hold for a second
15D9: 1A              LD      A,(DE)             ; From sprite data
15DA: D3 04           OUT     (SHFT_DATA),A      ; Write data to shift register
15DC: DB 03           IN      A,(SHFT_IN)        ; Read back shifted amount
15DE: 77              LD      (HL),A             ; Shifted sprite to screen
15DF: 23              INC     HL                 ; Adjacent cell
15E0: 13              INC     DE                 ; Next in sprite data
15E1: AF              XOR     A                  ; 0
15E2: D3 04           OUT     (SHFT_DATA),A      ; Write 0 to shift register
15E4: DB 03           IN      A,(SHFT_IN)        ; Read back remainder of previous
15E6: 77              LD      (HL),A             ; Write remainder to adjacent
15E7: E1              POP     HL                 ; Old screen coordinate
15E8: 01 20 00        LD      BC,$0020           ; Offset screen ...
15EB: 09              ADD     HL,BC              ; ... to next row
15EC: C1              POP     BC                 ; Restore count
15ED: 05              DEC     B                  ; All done?
15EE: C2 D7 15        JP      NZ,$15D7           ; No ... do all
15F1: E1              POP     HL                 ; Restore HL
15F2: C9              RET                        ; Done
```

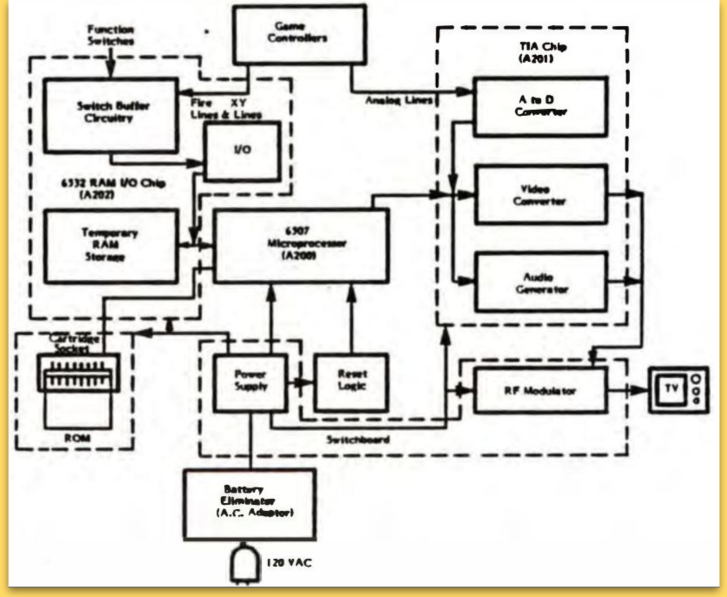
DrawSprite function in assembly

PROGRAMMING

Making an Emulator: Space Invaders on the Intel 8080

https://8080.cakers.io

ATARI 2600



Atari 2600 System Architecture

Atari 2600

But games had to be ported for at home play.

Specifications:

- Processor: MOS 6507 @1.19MHz
- RAM: 128 B
- Storage: 4 KB on ROM
- Video memory: none
- Resolution: 160 x 192

References

- Computer Archeology - Space Invaders
- Space Invaders – 30th Anniversary Developer Interview
- MITS Altair8800 - User documentation for emuStudio
- Atari 2600 field service manual by Bridal Association of America - Issuu

1980s

PROPRIETARY ENGINES

1980s

2-BIT
4-BIT
8-BIT
16-BIT
32-BIT
64-BIT
3D
HD

1980

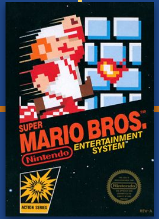
Pac-Man

NES
(8-bit)

1983

1985

Super Mario Bros.

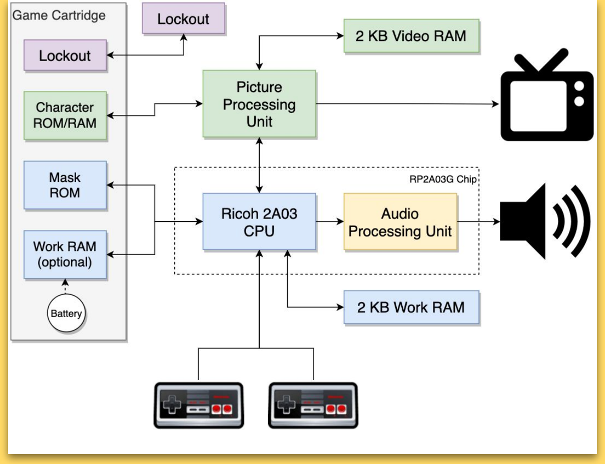


Sega Genesis
(16-bit)

1978

Nintendo Entertainment System (NES)

NES



NES System Architecture

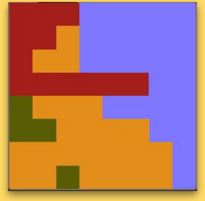
More subsystems are being added to the architecture.

Specifications:

- CPU: Ricoh2A03 @1.79 MHz
  - + Audio PU
  - + Picture PU
- RAM: 2 KB
- ROM: 50 KB (Program, Graphics)
  - Can be used for WRAM
- Video memory: 2 KB
- Resolution: 256x240

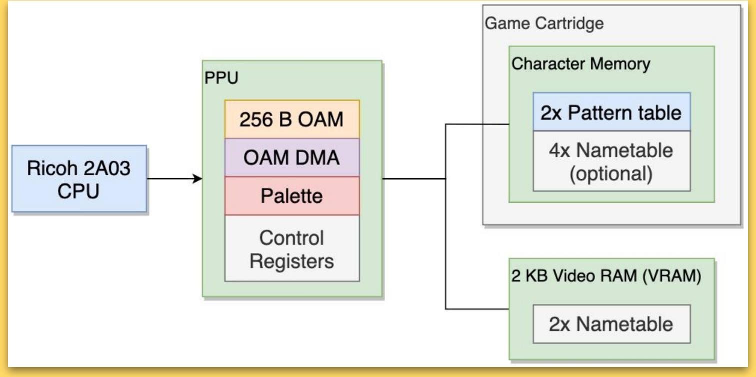
RENDERING

Mario Sprite Sheet



Mario Sprite

The sprite sheet can only have 8 sprites per scanline.
Each sprite is used by the PPU to fill a 8x8 pixel map called a tile.

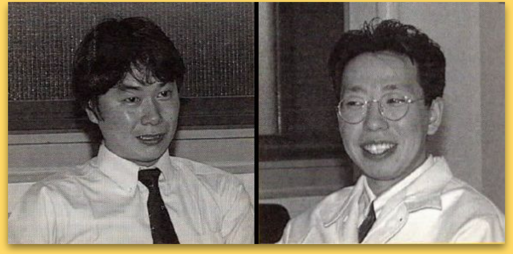


Graphics subsystem

Object Attribute Memory (OAM) specifies which tiles to put sprites in.

CPU and PPU work in unison to render the map and sprite frames to the Cathode Ray Tube gun.

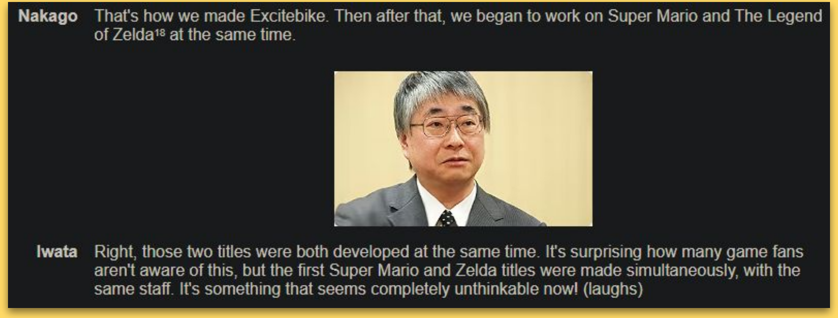
COLLABORATION



Designers of Super Mario Bros.
Shigeru Miyamoto and Takashi Tezuka

Work was divided amongst others like Koji K. being the Composer, and Toshihiko N. and others as programmers.

It took about 1.5 years to create through his own iterations and extensive testing.
Miyamoto brought the side-scrolling engine from Excitebike to Super Mario Bros.

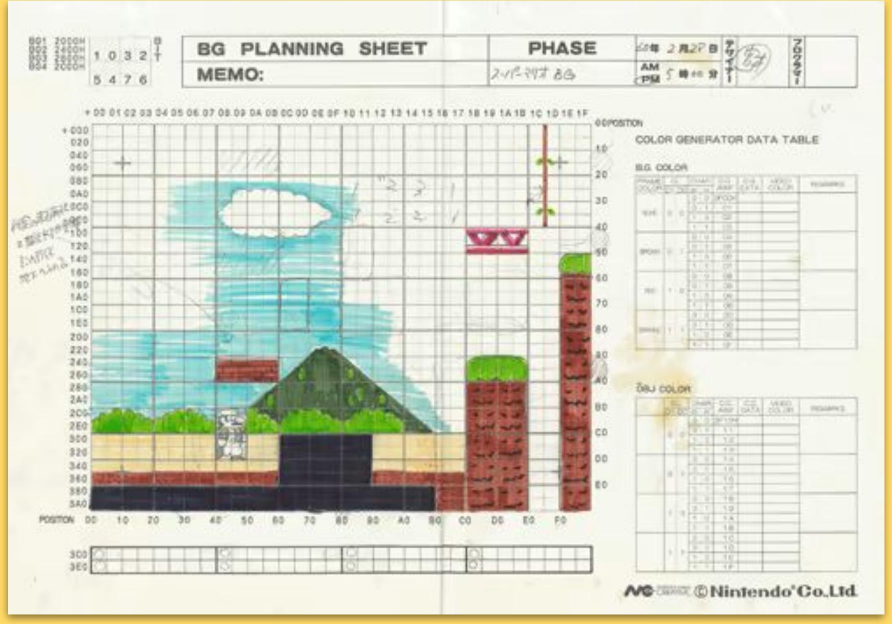


Nakago That's how we made Excitebike. Then after that, we began to work on Super Mario and The Legend of Zelda[18] at the same time.

Iwata Right, those two titles were both developed at the same time. It's surprising how many game fans aren't aware of this, but the first Super Mario and Zelda titles were made simultaneously, with the same staff. It's something that seems completely unthinkable now! (laughs)

Toshihiko Nakago interviewed about the reality of game development

LEVEL DESIGN



BG PLANNING SHEET

MEMO:

PHASE

スーパーマリオ BG

COLOR GENERATOR DATA TABLE

B.G. COLOR

OBJ COLOR

©Nintendo Co.,Ltd.

POSITION
00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
1 0 3 2
5 4 7 6

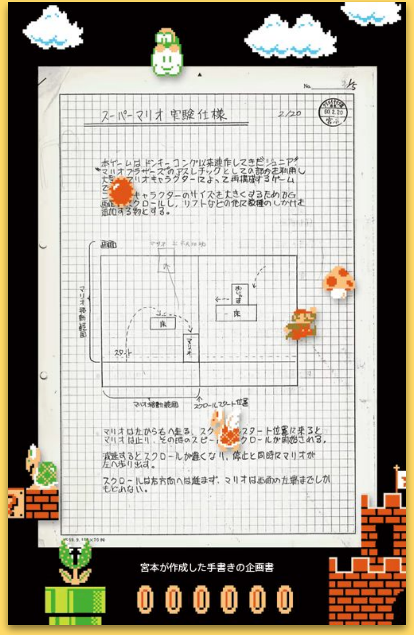
Designing the level pixel usage in Super Mario Bros.

IMPLEMENTING LEVEL DESIGNS

The Story of Super Mario Bros. 3

YouTube link

GAME DESIGN



No. 2/5

スーパーマリオ 実験仕様 2/20

60.2.20 宮本

本ゲームはドンキーコング以来連作してきたジュニア／マリオブラザーズのアスレチックとしての部分を利用し大型[illegible]マリオキャラクターによって再構成するゲームで[illegible]
こ[illegible]キャラクターのサイズを大きくするためBG画[illegible]クロールし、リフトなどの他に数種のしかけを追加する物とする。

画面
スタート
マリオ
床
マリオ移動範囲
スクロールスタート位置

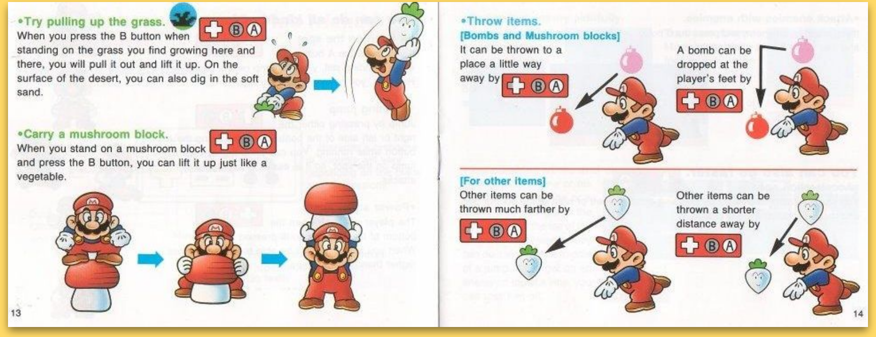
マリオは左から右へ走る。スク[illegible]スタート位置に来るとマリオは止り、その時のスピー[illegible]クロールが開始される。

減速するとスクロールが遅くなり、停止と同時にマリオが左へ歩り出す。

スクロールは右方向へは進まず、マリオは画面の左端までしかもどれない。

宮本が作成した手書きの企画書

Game Design of the jump



•Try pulling up the grass.

When you press the B button when standing on the grass you find growing here and there, you will pull it out and lift it up. On the surface of the desert, you can also dig in the soft sand.

•Carry a mushroom block.

When you stand on a mushroom block and press the B button, you can lift it up just like a vegetable.

13

•Throw items.

[Bombs and Mushroom blocks]

It can be thrown to a place a little way away by ✚ Ⓑ Ⓐ

A bomb can be dropped at the player's feet by ✚ Ⓑ Ⓐ

[For other items]

Other items can be thrown much farther by ✚ Ⓑ Ⓐ

Other items can be thrown a shorter distance away by ✚ Ⓑ Ⓐ

14

Book instructions on the pick up mechanic

Building on an installment makes it easy to explore new ways to attract new players.

Specially when competition is near...

GRAPHICS

NES Scrolling Basics featuring Super Mario Bros. - Behind the Code

YouTube link

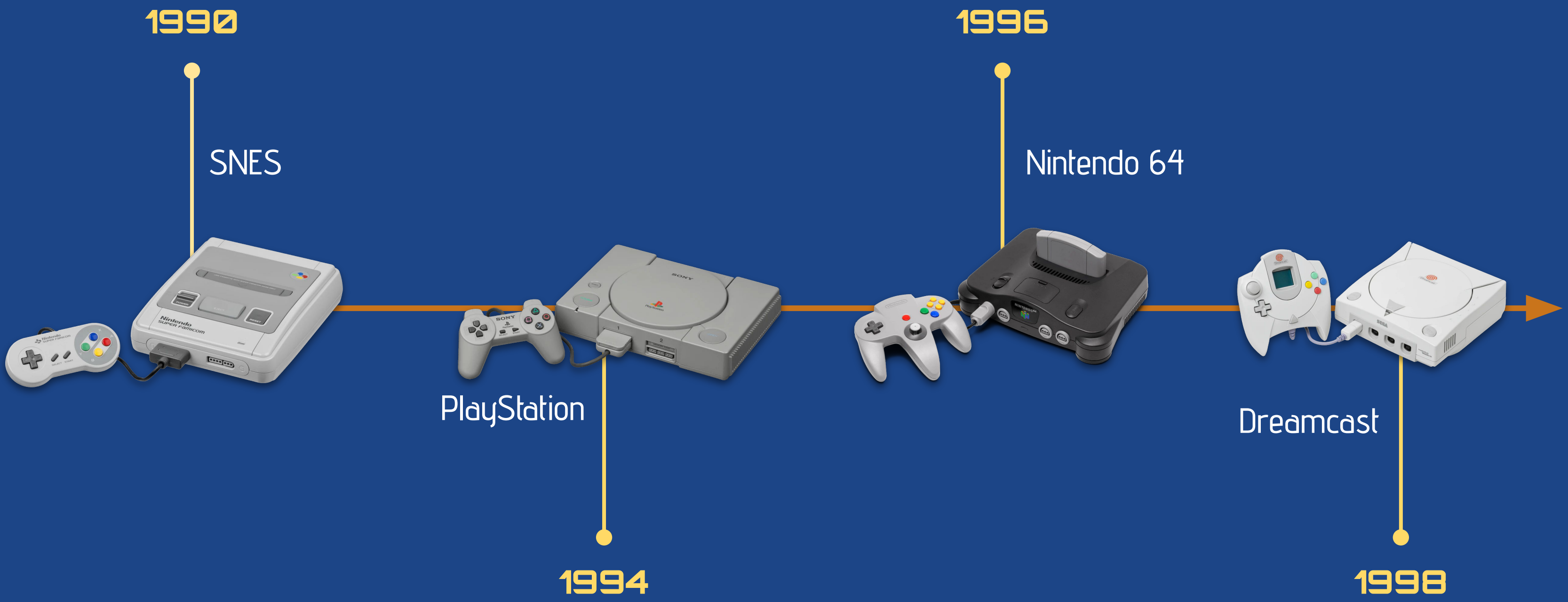
References

- Nintendo Entertainment System (NES) Architecture | A Practical Analysis
- Iwata Asks - New Super Mario Bros: Volume 2 - Page 1
- Iwata Asks - Volume 5 : Original Super Mario Developers - Page 4

1990s

3D Graphics

1990s



1990

SNES

1994

PlayStation

1996

Nintendo 64

1998

Dreamcast

DOOM

Doom cover for PC

Developed by id Software and released in 1993 for DOS.

Co-founders John Carmack (lead programmer) and John Romero (designer/programmer), and Tom Hall (game designer) and Adrian Carmack (artist).

Doom was developed in 15 months with most of the team from Wolfenstein 3D and a first example of 3D graphics (or 2.5D with 2D enemies).

PC (1993)

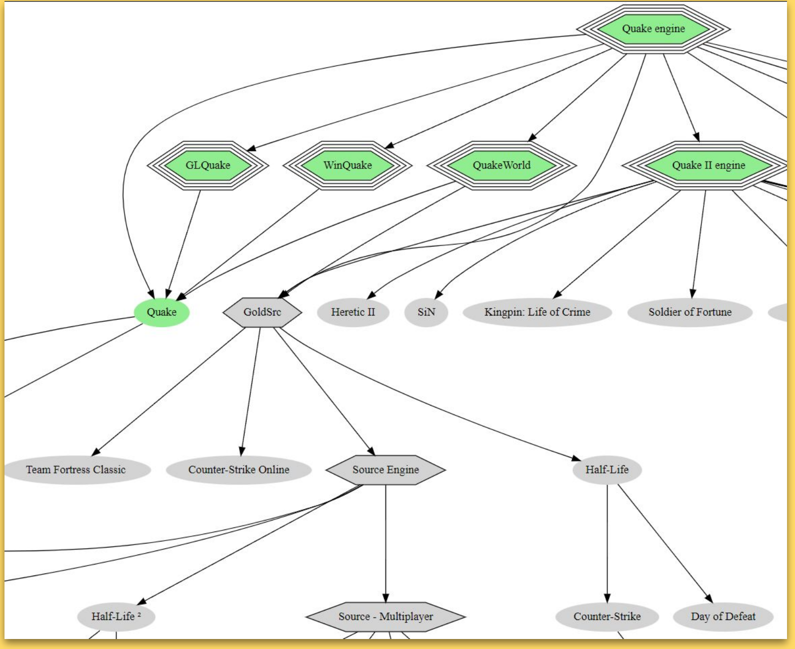
IBM PS/2 running Doom

It was published for DOS first.

Requirements:

- Typically a IBM PS/2 Series
- CPU:
  - Intel 80386 (386) @25 MHz
  - Intel 80486 (486) @33 MHz
- RAM: 4 MB-8MB
- Graphics: VGA adapted card
- Ex: XGA-2 with 1 MB VRAM:
  - 1024x768 pixels with 256 colors
- Storage: 25 MB to install, 10 MB to run
- Sound: Sound Blaster

GAME ENGINES (1999)

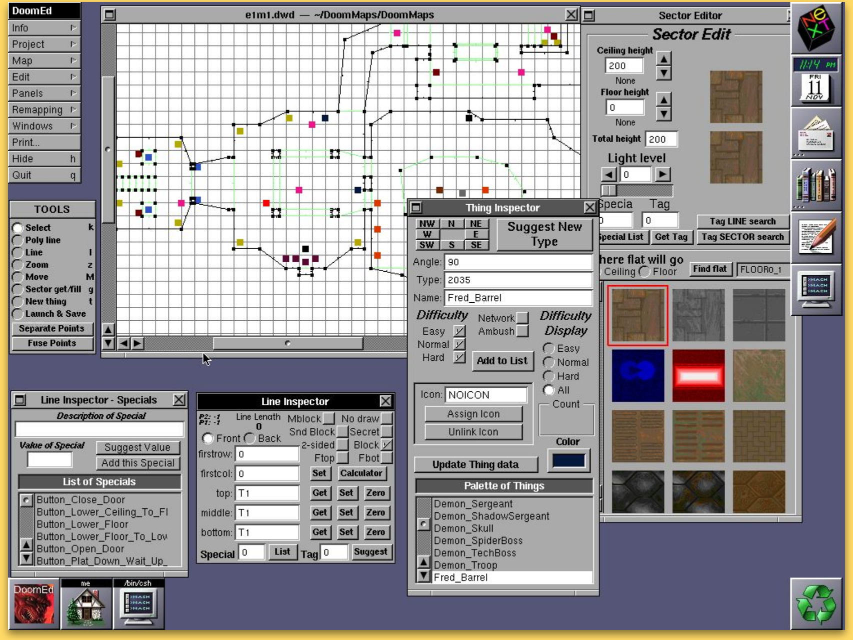


Part of the engine reusability tree

Doom's engine went to be reused as Quake engine which when released under GNU General Public License, inspired many engines still used Today.

At this point, independent Game Engines became a thing.

LEVEL EDITOR



Sector Editor:
Ceiling height 200
None
Floor height 0
None
Total height 200
Light level 0
Specia 0 Tag 0
FLOOR0_1

Thing Inspector:
Angle: 90
Type: 2035
Name: Fred_Barrel
Difficulty: Easy ☑ Normal ☑ Hard ☑ Network □ Ambush □
Icon: NOICON

Palette of Things:
Demon_Sergeant
Demon_ShadowSergeant
Demon_Skull
Demon_SpiderBoss
Demon_TechBoss
Demon_Troop
Fred_Barrel

Line Inspector:
Line Length 0
firstrow: 0
firstcol: 0
top: T1
middle: T1
bottom: T1
Special 0
Tag 0

List of Specials:
Button_Close_Door
Button_Lower_Ceiling_To_Fl
Button_Lower_Floor
Button_Lower_Floor_To_Low
Button_Open_Door
Button_Plat_Down_Wait_Up_

Doom Editor

Objective-C was used to program the Doom engine.

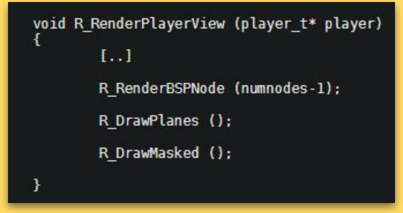
Wolfenstein 3D used Grids with ray casting to determine what to render.

Doom went for BSP Trees to represent partitions of what to render and its order.

DOOM ENGINE

BSP rendering sequence

Player location determines what to render:

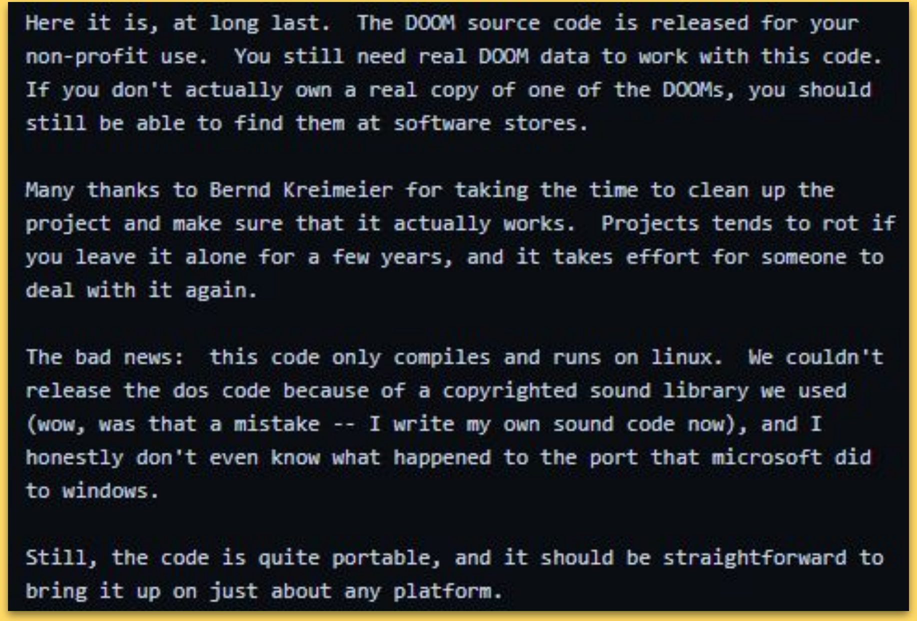


```
void R_RenderPlayerView (player_t* player)
{
        [..]

        R_RenderBSPNode (numnodes-1);

        R_DrawPlanes ();

        R_DrawMasked ();

}
```

Sample code to render player view

Full article

RELEASING THE CODE



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.

The bad news: this code only compiles and runs on linux. We couldn't release the dos code because of a copyrighted sound library we used (wow, was that a mistake -- I write my own sound code now), and I honestly don't even know what happened to the port that microsoft did to windows.

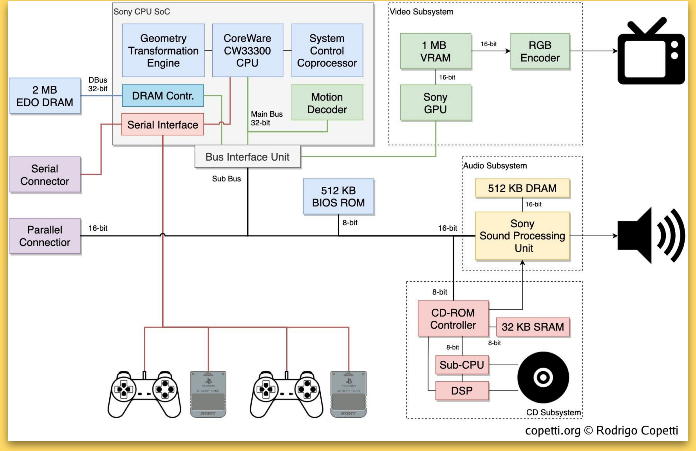
Still, the code is quite portable, and it should be straightforward to bring it up on just about any platform.

John Carmack note on releasing the code

GitHub source code

PlayStation PORT (1995)

PlayStation

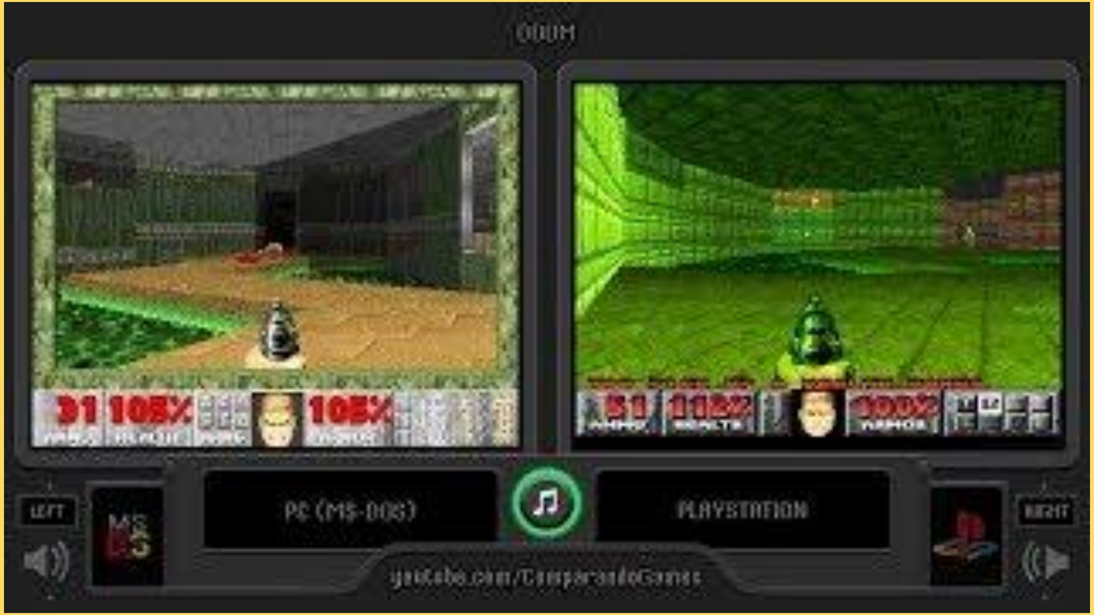


PlayStation System Architecture

Specifications:

- CPU: Sony CXD8530BQ @33.87 MHz
- Audio subsystem
- Video:
  - GPU + 1 MB VRAM
- RAM: 2 MB
- BIOS
- Resolution: 256x240

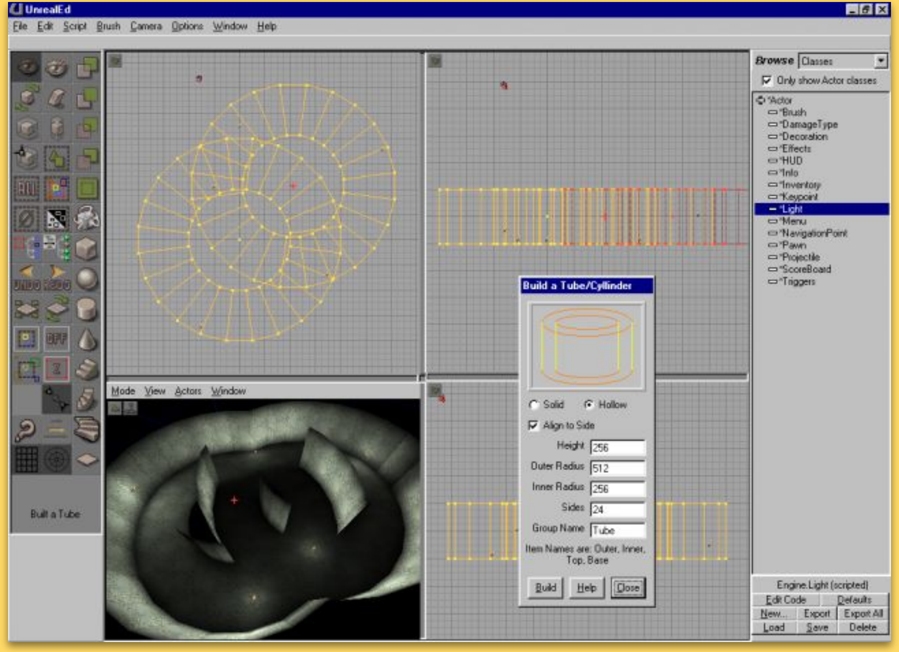
IMPLEMENTING LEVEL DESIGNS



PC vs PlayStation port

YouTube link

GAME EDITORS



Build a Tube/Cyllinder

○ Solid ● Hollow

☑ Align to Side

Height 256

Outer Radius 512

Inner Radius 256

Sides 24

Group Name Tube

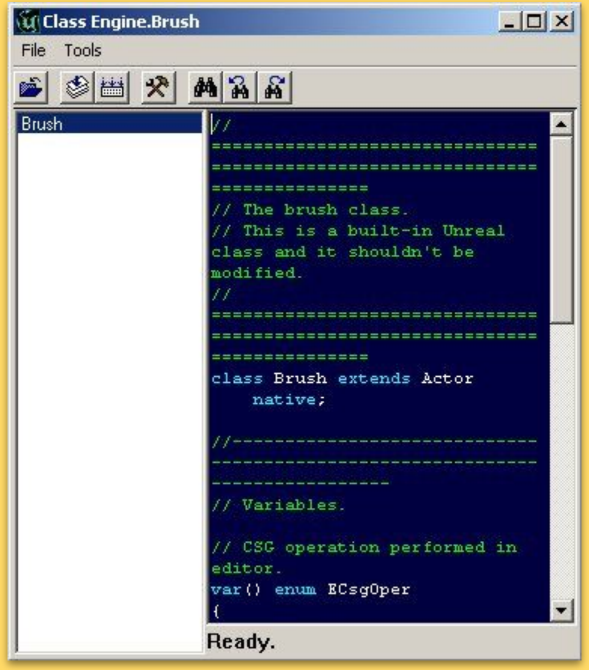
Item Names are: Outer, Inner, Top, Base

First Unreal Editor

This is Tom Sweeney's (Founder of Unreal) editor for Unreal Engine. It was written in Visual Basic.

The editor can do Real Time BSP and continue editing maps. There is the possibility for volumetric lighting.

GENERAL GAME ENGINES

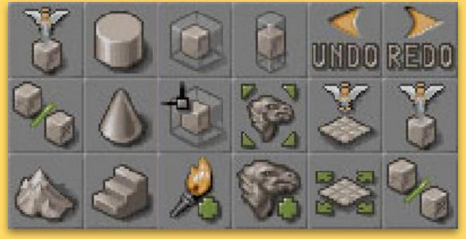


```
//
===============================
===============================
===============
// The brush class.
// This is a built-in Unreal
class and it shouldn't be
modified.
//
===============================
===============================
===============
class Brush extends Actor
    native;

//-----------------------------
-------------------------------
-----------------

// Variables.

// CSG operation performed in
editor.
var() enum ECsgOper
{
```

Sample UnrealScript – Today replaced with C++

UnrealScript was also created to interact with the Engine in order to build games.

The Unreal Editor would call these classes for simplicity.



UnrealEd widgets facilitating game dev

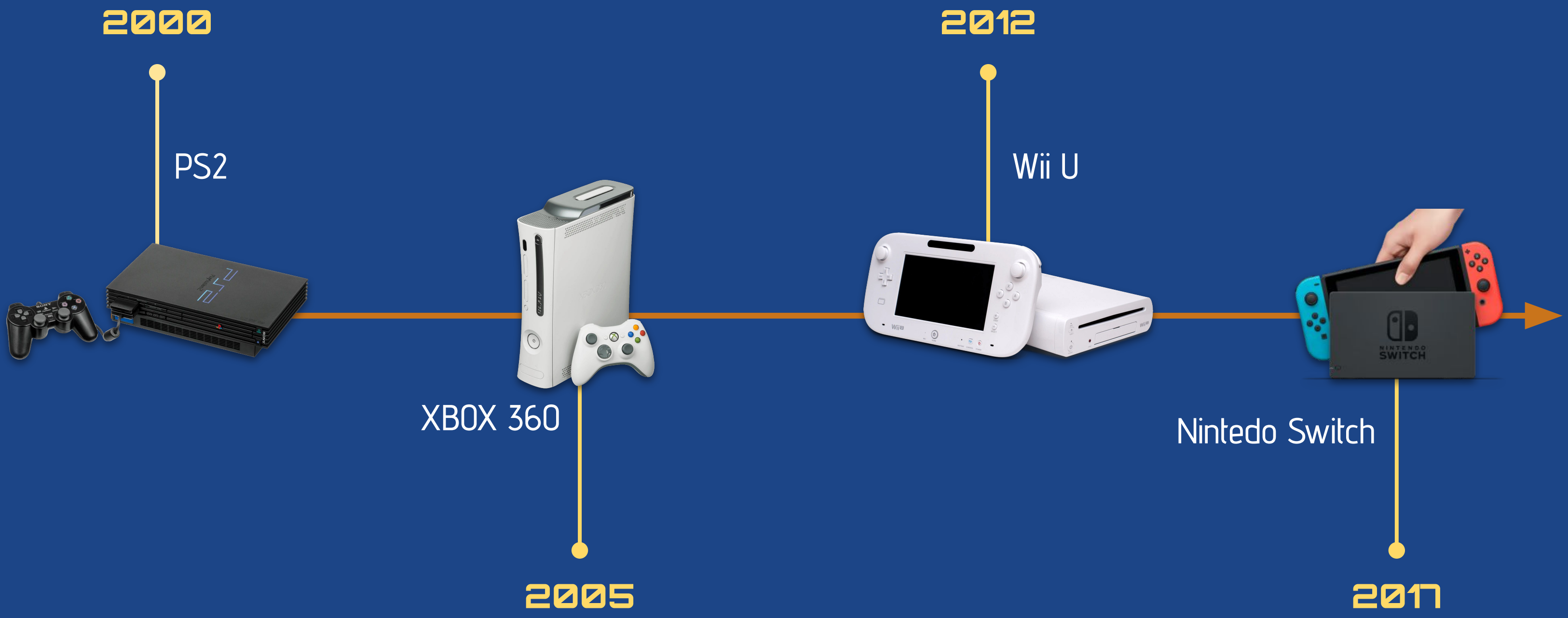
References

- The XGA Graphics Chip | OS/2 Museum
- Original Doom system requirements (from my 25 year old retail box) : r/gaming
- Development of Doom - The Doom Wiki at DoomWiki.org
- Monsters from the Id: The Making of Doom
- PlayStation Architecture | A Practical Analysis
- Каким был первый Unreal Editor

2000s

(PLAY || F5)

2000s-Today



2000

PS2

XBOX 360

2005

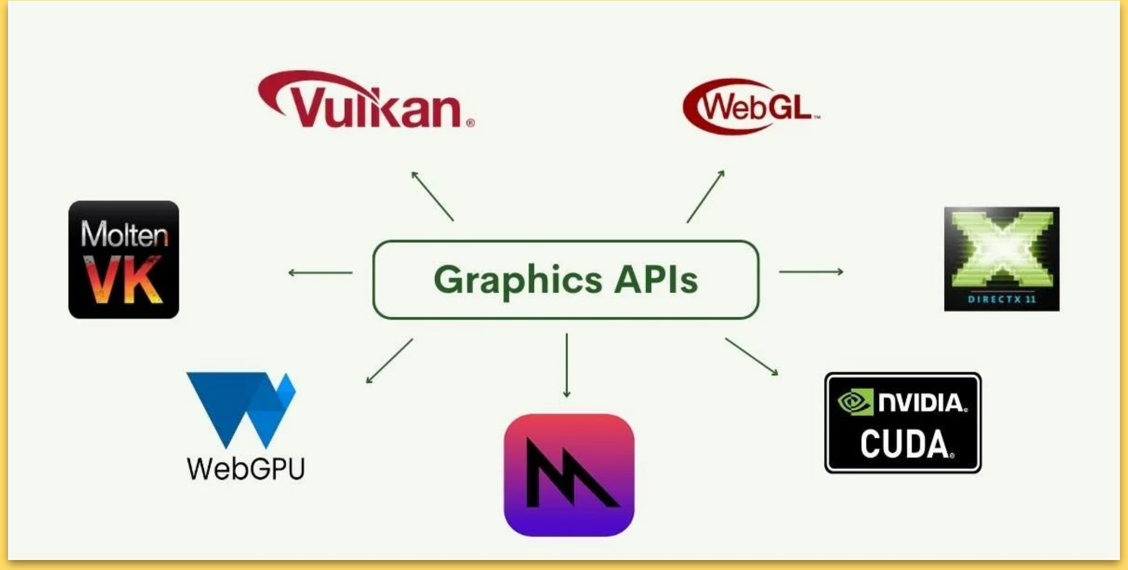
2012

Wii U

Nintedo Switch

2017

3D GRAPHICS LIBRARIES



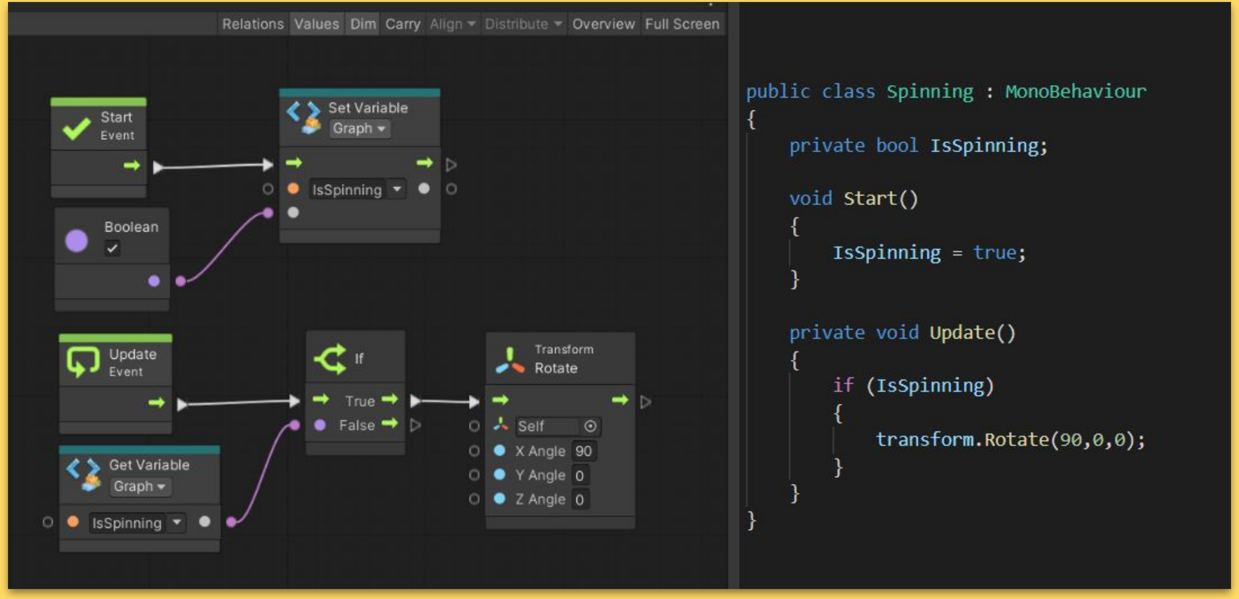
Some Graphics Libraries

COMMERCIAL GAME ENGINES

unity
UNREAL ENGINE
GODOT Game engine
CRYENGINE®
GameMaker™

Some Game Engines

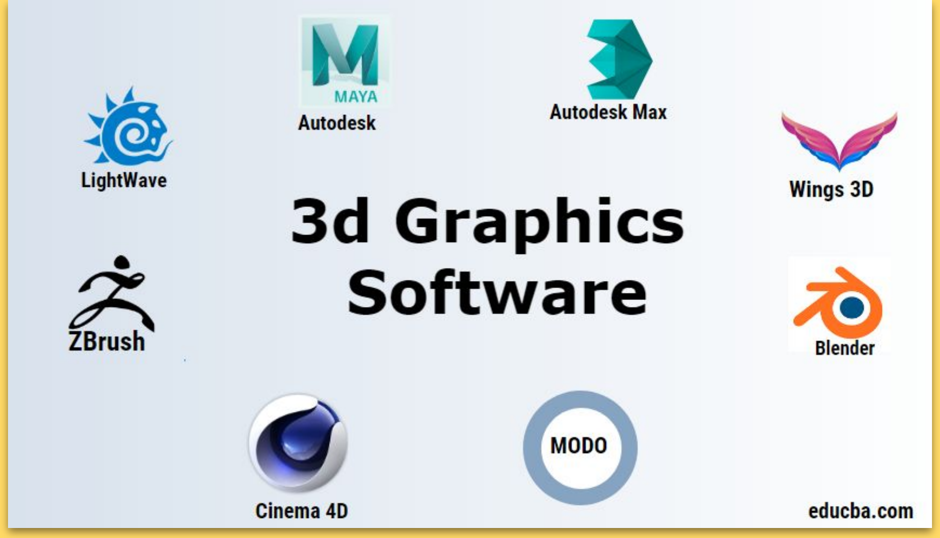
VISUAL SCRIPTING



```
public class Spinning : MonoBehaviour
{
    private bool IsSpinning;

    void Start()
    {
        IsSpinning = true;
    }

    private void Update()
    {
        if (IsSpinning)
        {
            transform.Rotate(90,0,0);
        }
    }
}
```

Visual Scripting in Unity

3D GRAPHICS TOOLS



Some Graphics Tools

AUDIO MIDDLEWARE

fmod
Wwise
elias
ADAPTIVE MUSIC
CRIWARE
Fabric

Some Audio Middleware

References

- Graphics API's Explained : r/GraphicsProgramming
- VANAS | Top 5 Video Game Engines
- https://learn.unity.com/tutorial/about-unity-visual-scripting
- 3d Graphics Software | Learn the Top Software of 8 3d Graphics
- Audio Middleware: Why would I want it in my game?

GAMING STUDIOS

An industry perspective

NDA

CONFIDENTIALITY AGREEMENT

The Confidentiality Agreement is made by and between:

(1st Par

And

Every Video Game Studio's first contract

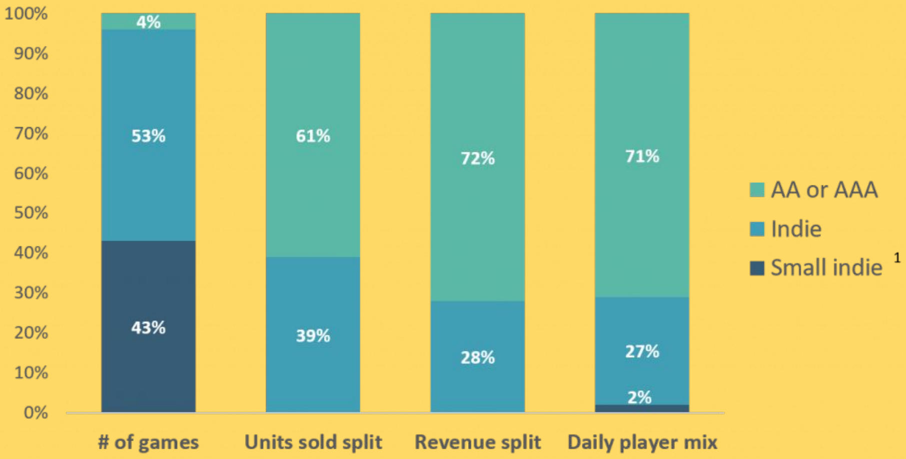
INDIE vs AA vs AAA

Characteristics	Indie	AAA	AAA
Budget	~1K to $1M	Several millions	Over $50 million
Team Size	1-5 people	Below 50 people	Large teams, in the 100s
Dedicated Publisher	Often lacks a dedicated publisher	Backed by a publisher with more creative freedom	Typically published by established companies
Production Values	Smaller scale, emphasis on mechanics	Good production values, less famous actors	High production values, famous voice actors
Threshold for Success	Low, due to lower development budgets	Between AAA and Indie games	High, often selling over 2 million copies

Video Game type comparison

REALITY OF GAMES

Distribution of Steam Games Based on Publisher Type



[1] – Indie games with <300 units sold

Distribution table

INDUSTRY ROLES

Engineering

Gameplay programmers
Engine programmers

Production

Producers
Coordinators

QA

Testers
Test Engineers

Design

Game Design
Level Design
Creative/Game Direction

Art

Concept Art
Environment Art
Character Art

Narrative

Storyboard
Dialogue

Technical Art

Visual Effects Artist
Technical Artist (ex: Rigging)

Audio

Composer
Sound Designer

Animation

Animator
Technical Animator

References

- A Comparison Indie, AA vs AAA Games
- Indie, AA, and AAA Games: The Ultimate Guide
- Indie games make up 40% of all units sold on Steam

FULL CIRCLE

Cuphead

HUMAN CREATIVITY

Cuphead trailer

HAND DRAWN ANIMATIONS

Hand drawn animations

YouTube link

LIVE MUSIC

Live music composition

YouTube link

UNITY PROJECT

Demo of the Unity Cuphead project

Playable WebGL port

GitHub source code

MAKING GAMES

Making of Cuphead

YouTube link

CONCLUSION

What to take away?

HAVE FUN!

Making games is now more accessible than ever.

Start simple, iterate a lot, find your creativity, and don't give up!

Be part of the adventure.