Using WASM to port games to the browser

The BlazorDoom use case

DEVOXX United Kingdom

How it started

How it's going

Yassine
Benabbas

@yostane

DevRel / Teacher

Video game collector

😍 Kotlin, WASM, or any fancy technology

WORLDLINE

#TechAtWorldline

WORLDLINE

We design **payments technology** that powers the growth of millions of businesses around the world.

Handling 150+ payment methods

7000+ engineers in over 40 countries

Managing 43+ billion transactions per year

#TechAtWorldline

blog.worldline.tech

Prolog

WebAssembly

Web Assembly (WASM)

- Portable binary instruction format
- Initially targeted for web browsers

Web Assembly (WASM)

- Portable binary instruction format
- Initially targeted for web browsers

- Faster than JS on compute intensive tasks
- Many programming languages compile to WASM

wasm file

```
1  00 61 73 6d 01 00 00 00  01 05 01 60 00 01 7f 03  |.asm.......`....|
2  02 01 00 07 16 01 12 67  65 74 55 6e 69 76 65 72  |.......getUniver|
3  73 61 6c 4e 75 6d 62 65  72 00 00 0a 06 01 04 00  |salNumber.......|
4  41 2a 0b 00 0a 04 6e 61  6d 65 02 03 01 00 00     |A*....name.....|
```

wasm file

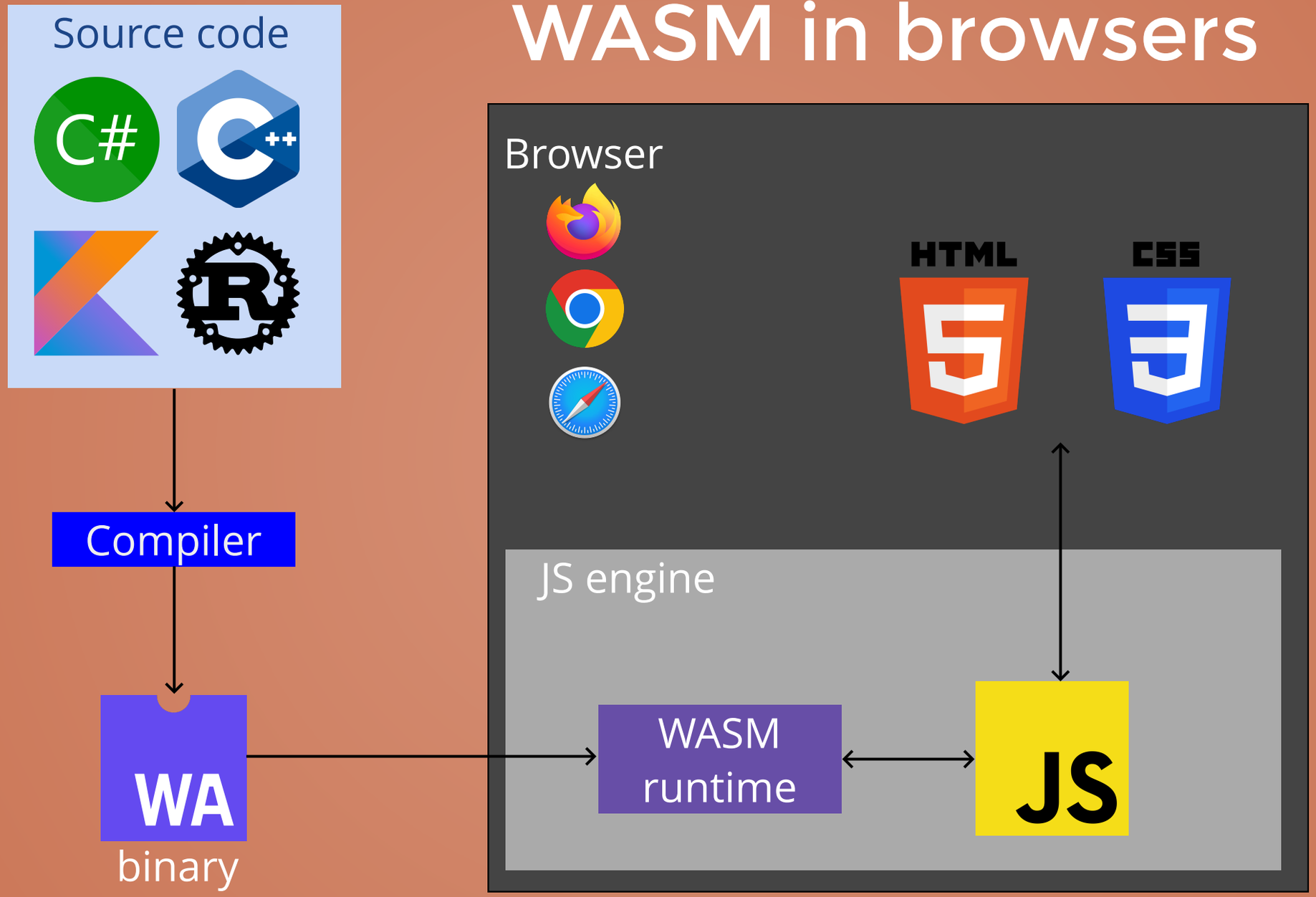
```
1  00 61 73 6d 01 00 00 00  01 05 01 60 00 01 7f 03  |.asm.......`....|
2  02 01 00 07 16 01 12 67  65 74 55 6e 69 76 65 72  |.......getUniver|
3  73 61 6c 4e 75 6d 62 65  72 00 00 0a 06 01 04 00  |salNumber.......|
4  41 2a 0b 00 0a 04 6e 61  6d 65 02 03 01 00 00     |A*....name.....|
```


https://github.com/WebAssembly/wabt

wat format (human readable)

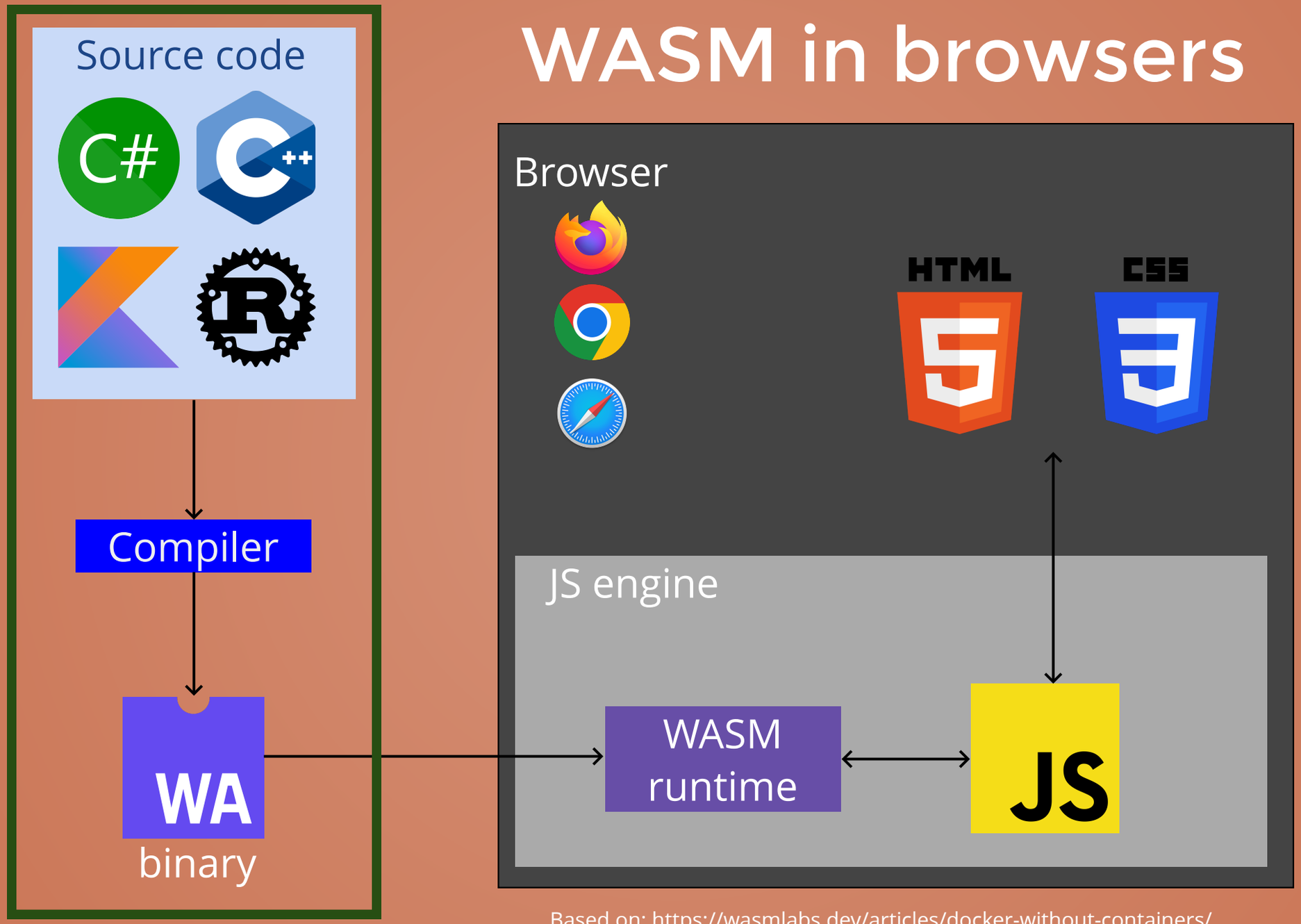
```
1  (module
2    (func (result i32)
3      (i32.const 42)
4    )
5    (export "getUniversalNumber" (func 0))
6  )
```

WASM in browsers



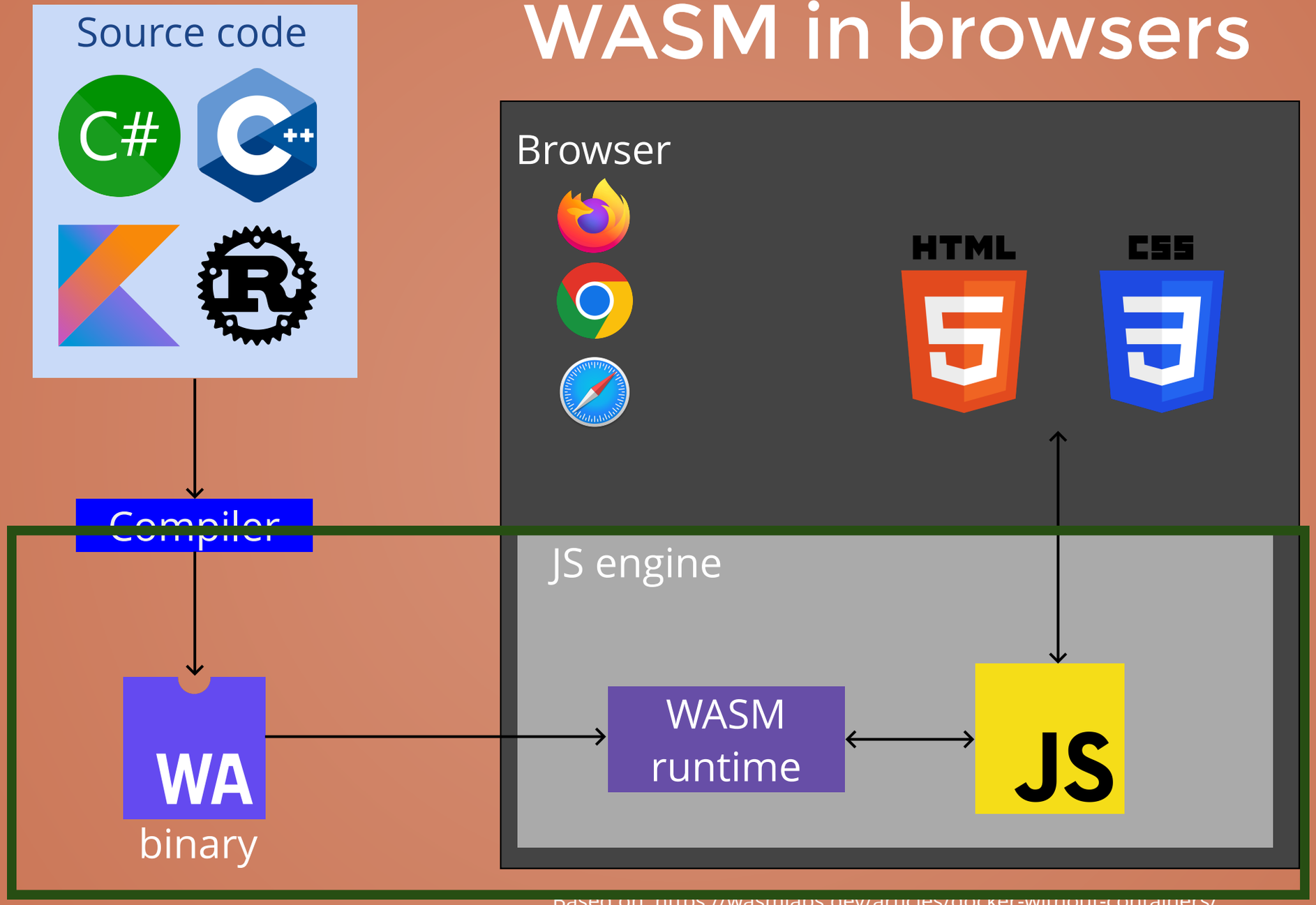
Based on: https://wasmlabs.dev/articles/docker-without-containers/

WASM in browsers



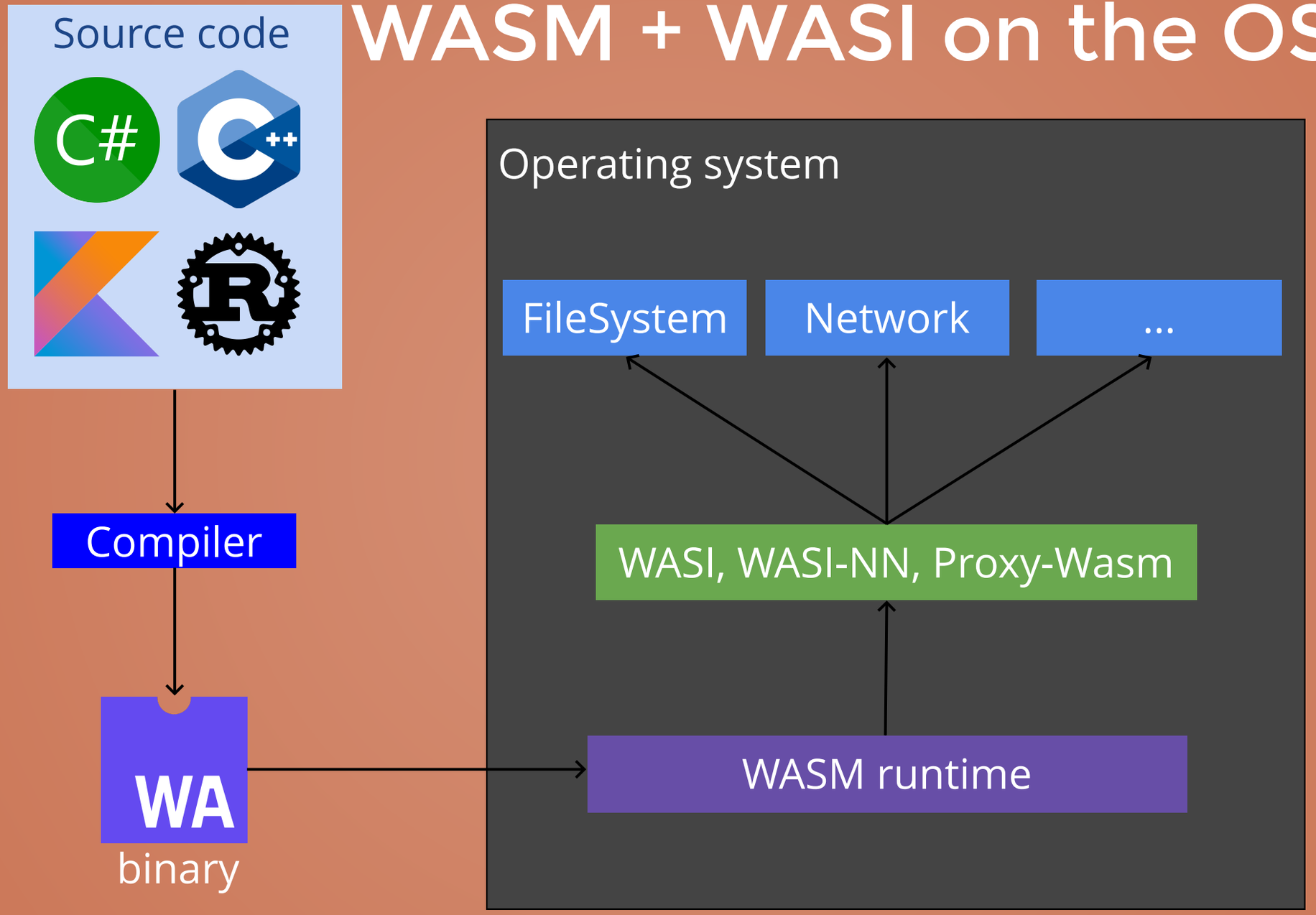
Based on: https://wasmlabs.dev/articles/docker-without-containers/

WASM in browsers



Based on: https://wasmlabs.dev/articles/docker-without-containers/

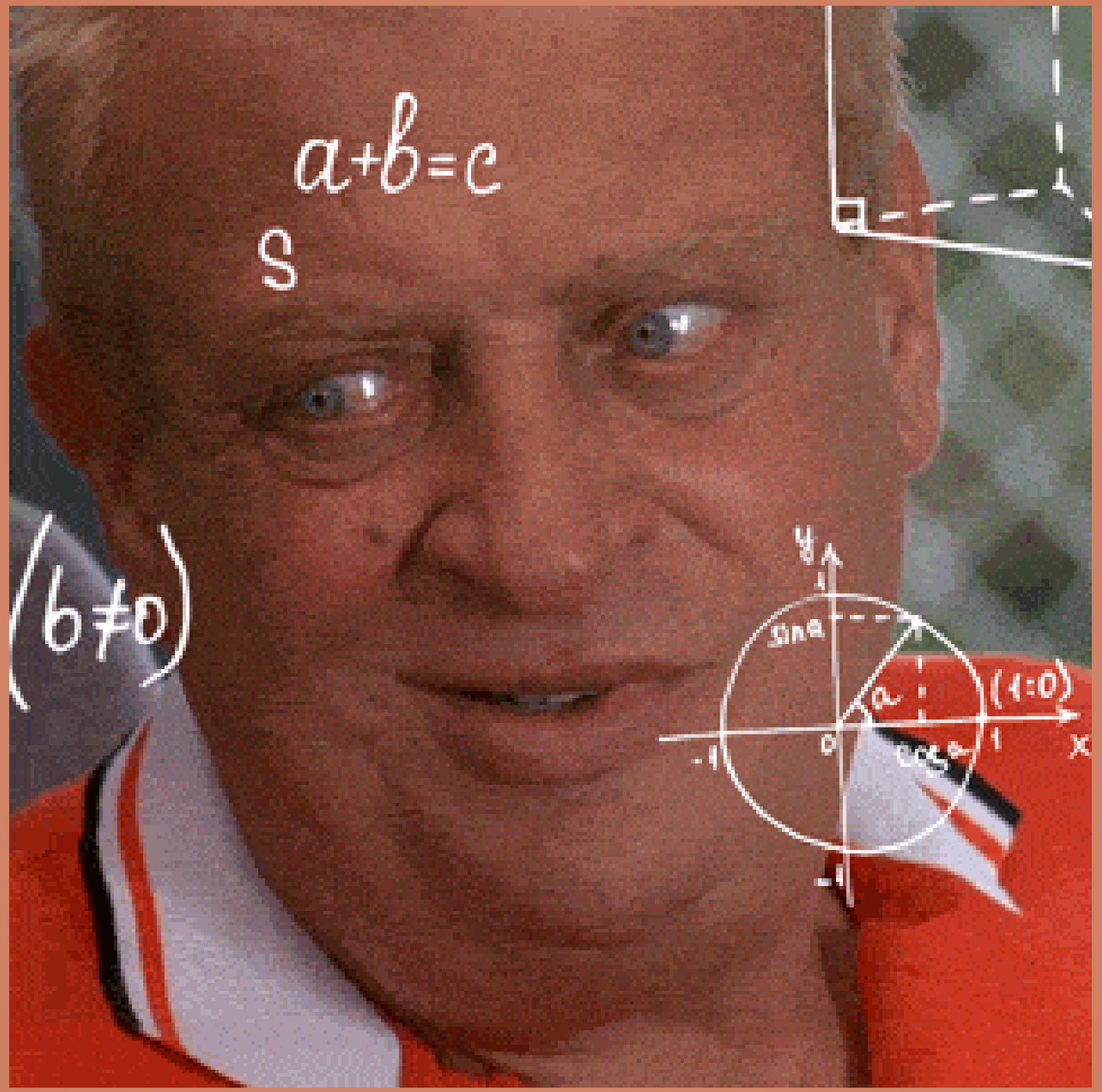
WASM + WASI on the OS



Based on: https://wasmlabs.dev/articles/docker-without-containers/

WASM on the browser

- A lot of languages compile to WASM
- JS interop



Need to do something related to WASM

Chapter 1

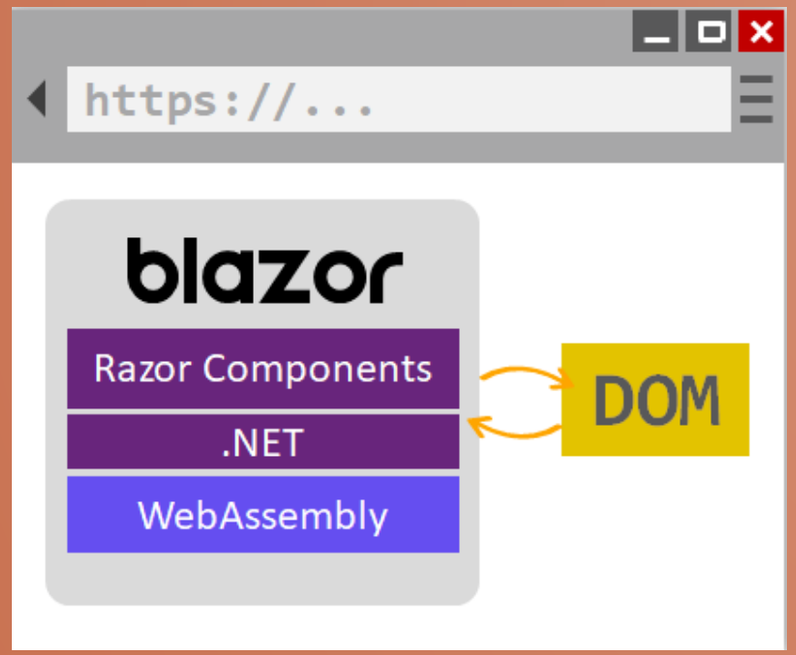
.Net and WASM

.NET on the browser

- .NET: C# OSS cross-platform framework

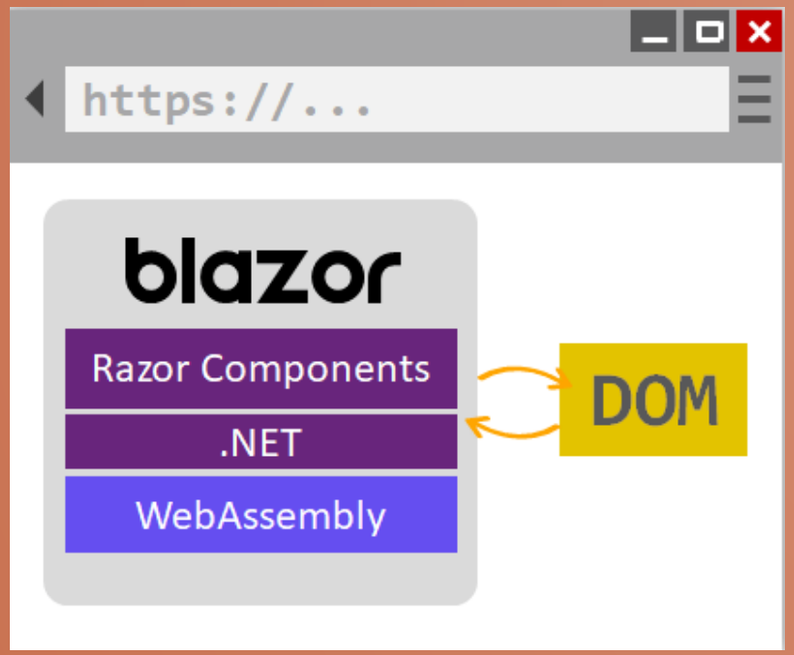
.NET on the browser

- .NET: C# OSS cross-platform framework
- .Net 5 (2020): Blazor WASM
  - Component-based framework
  - .Net runtime compiled into WASM

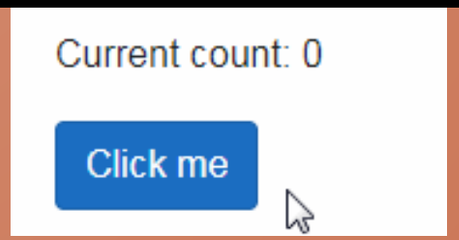


.NET on the browser

- .NET: C# OSS cross-platform framework
- .Net 5 (2020): Blazor WASM
  - Component-based framework
  - .Net runtime compiled into WASM




```razor
@page "/counter"

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;
    private void IncrementCount() { currentCount++; }
}
```



.NET 7+ wasm-tools

- Call .Net from JS and vice-versa
- Basis of Blazor WASM

```C#
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

```JS
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

.NET 7+ wasm-tools

- Call .Net from JS and vice-versa
- Basis of Blazor WASM

```C#
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

```JS
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

.NET 7+ wasm-tools

- Call .Net from JS and vice-versa
- Basis of Blazor WASM

```C#
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

```JS
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

.NET 7+ wasm-tools

- Call .Net from JS and vice-versa
- Basis of Blazor WASM

Blazor WASM

💪

Webapp or front dev

wasm-tools

💪

general purpose

C#

```csharp
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

JS

```javascript
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

Let's port a .NET game !


CHAMP 2
MANAGEDOOM

Game porting

- Make a game run in platforms other than its original ones
- By rewriting / adapting the source code for the new platform(s)
- Not porting: virtual machine or emulator

Game porting

- Make a game run in platforms other than its original ones
- By rewriting / adapting the source code for the new platform(s)
- Not porting: virtual machine or emulator

MVG's video is a great source of inspiration

- Released in 1993 for DOS
- One the most successful First Person Shooters
- Has two parts:
  - **engine: Game logic + I/O**
  - WAD file: all assets and maps

🌟 Doom is portable by design 🌟

Ported to a LOT of platforms

http://mrglitchsreviews.blogspot.com/2012/09/doom-console-ports.html

25 official licensed ports https://www.thegamer.com/doom-how-many-platforms-ports-consoles

https://www.link-cable.com/top-10-weird-doom-ports/

Source: https://www.yahoo.com/news/1995-bill-gates-gave-crazy-180900044.html

ManagedDoom

- **.NET** Port of LinuxDoom (official source code)
- ManagedDoom V1 Uses SFML (graphics + audio + input)
  - V2 uses silk.net (released 27 Dec 2022)
- My work is a fork of **ManagedDoom V1**

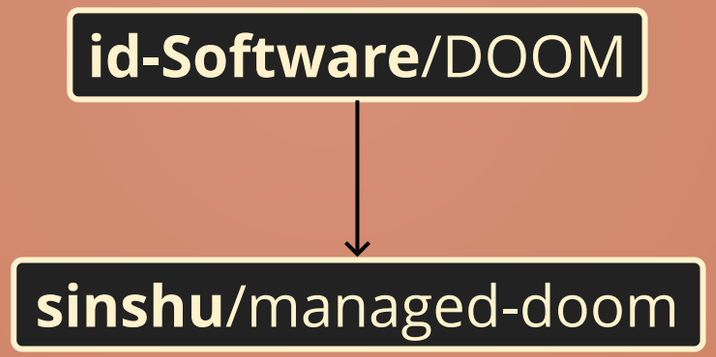
ManagedDoom

- **.NET** Port of LinuxDoom (official source code)
- ManagedDoom V1 Uses SFML (graphics + audio + input)
  - V2 uses silk.net (released 27 Dec 2022)
- My work is a fork of **ManagedDoom V1**

id-Software/DOOM

ManagedDoom

- **.NET** Port of LinuxDoom (official source code)
- ManagedDoom V1 Uses SFML (graphics + audio + input)
  - V2 uses silk.net (released 27 Dec 2022)
- My work is a fork of **ManagedDoom V1**



ManagedDoom

- **.NET** Port of LinuxDoom (official source code)
- ManagedDoom V1 Uses SFML (graphics + audio + input)
  - V2 uses silk.net (released 27 Dec 2022)
- My work is a fork of **ManagedDoom V1**

id-Software/DOOM

↓

sinshu/managed-doom

↓

yostane/MangedDoom-Blazor

CHAPTER 3

MAKING THE PORT

Game loop in C#

```
while (waitForNextFrame()){
  const input = getPlayerInput();
  const { frame, audio }
              = UpdateGameState(input, WAD);
  render(frame);
  play(audio);
}
```

Game loop in JS

```
1  function gameLoop(){
2    if (canAdvanceFrame()){
3      const input = getPlayerInput();
4      const { frame, audio }
5                  = UpdateGameState(input, WAD);
6      render(frame);
7      play(audio);
8    }
9    requestAnimationFrame(gameLoop);
10 }
11 requestAnimationFrame(gameLoop);
```

Game loop in JS

```
1  function gameLoop(){
2    if (canAdvanceFrame()){
3      const input = getPlayerInput();
4      const { frame, audio }
5                  = UpdateGameState(input, WAD);
6      render(frame);
7      play(audio);
8    }
9    requestAnimationFrame(gameLoop);
10 }
11 requestAnimationFrame(gameLoop);
```

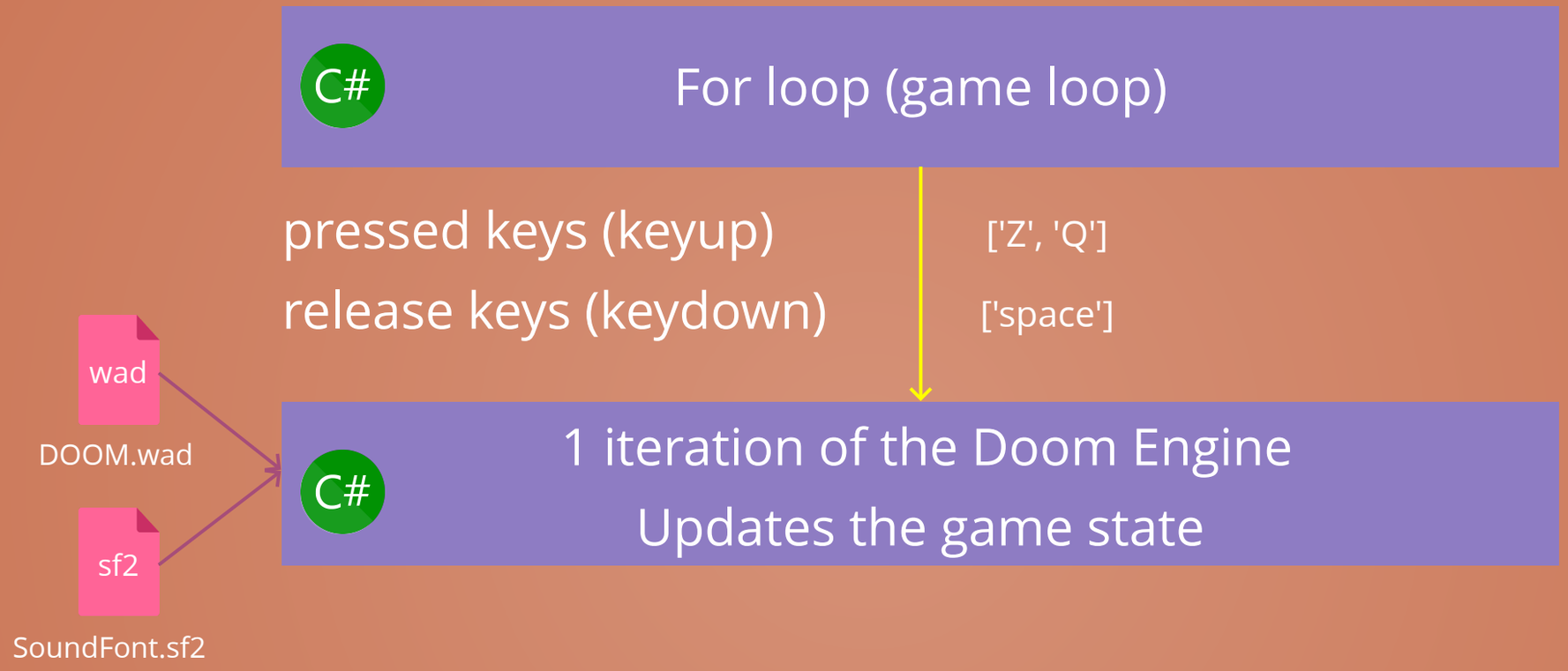
Game loop in JS

```
1  function gameLoop(){
2    if (canAdvanceFrame()){
3      const input = getPlayerInput();
4      const { frame, audio }
5                  = UpdateGameState(input, WAD);
6      render(frame);
7      play(audio);
8    }
9    requestAnimationFrame(gameLoop);
10 }
11 requestAnimationFrame(gameLoop);
```

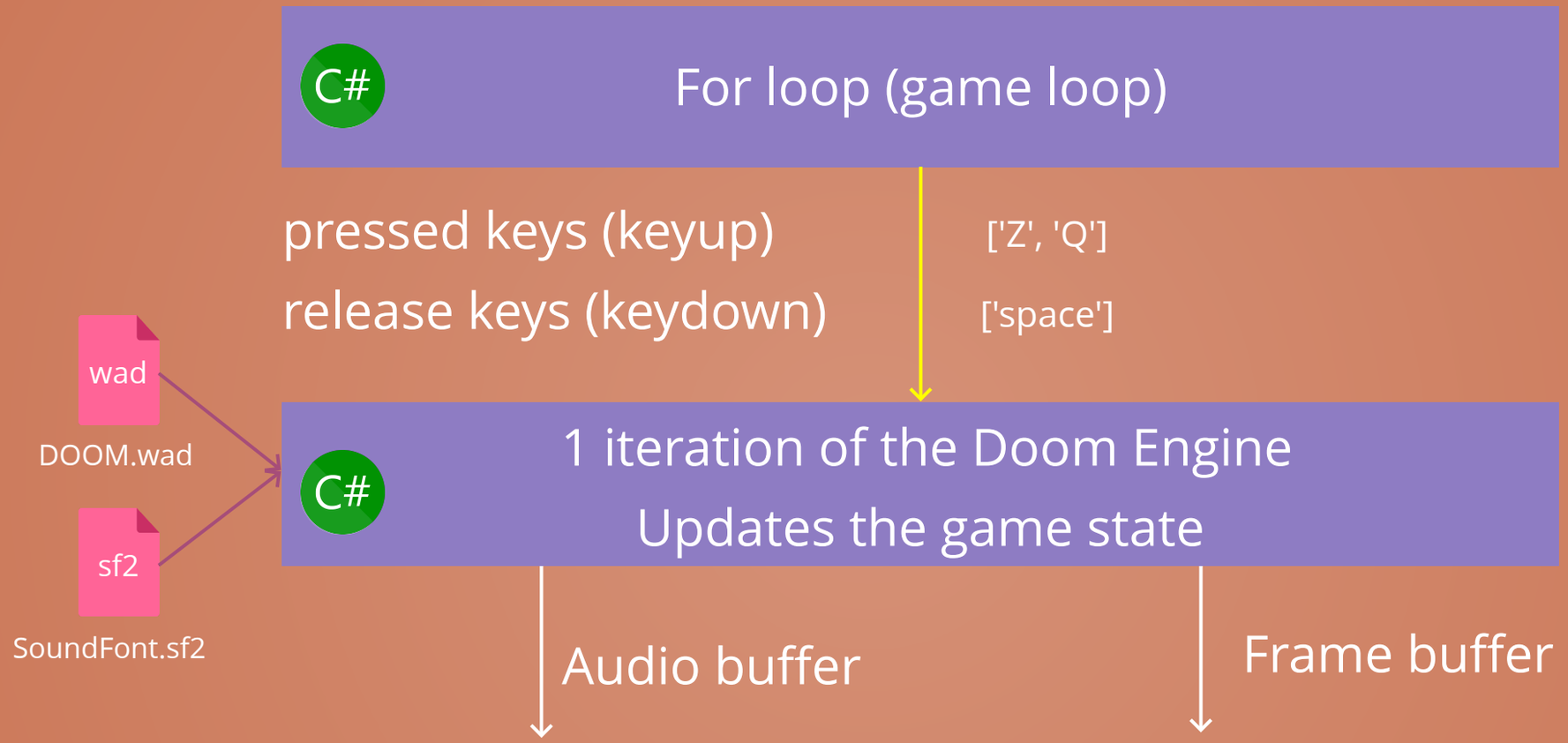
ManagedDoom V1 architecture

ManagedDoom V1 architecture

ManagedDoom V1 architecture



ManagedDoom V1 architecture



ManagedDoom V1 architecture

C# For loop (game loop)

pressed keys (keyup) ['Z', 'Q']

release keys (keydown) ['space']

wad

DOOM.wad

sf2

SoundFont.sf2

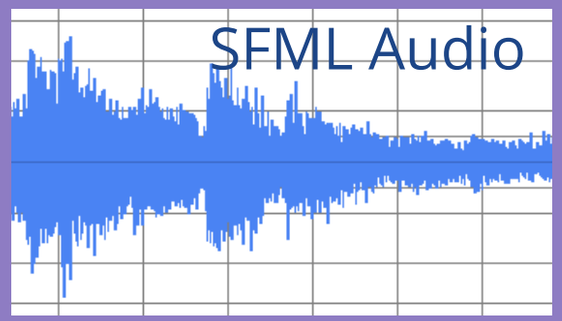
C# 1 iteration of the Doom Engine

Updates the game state

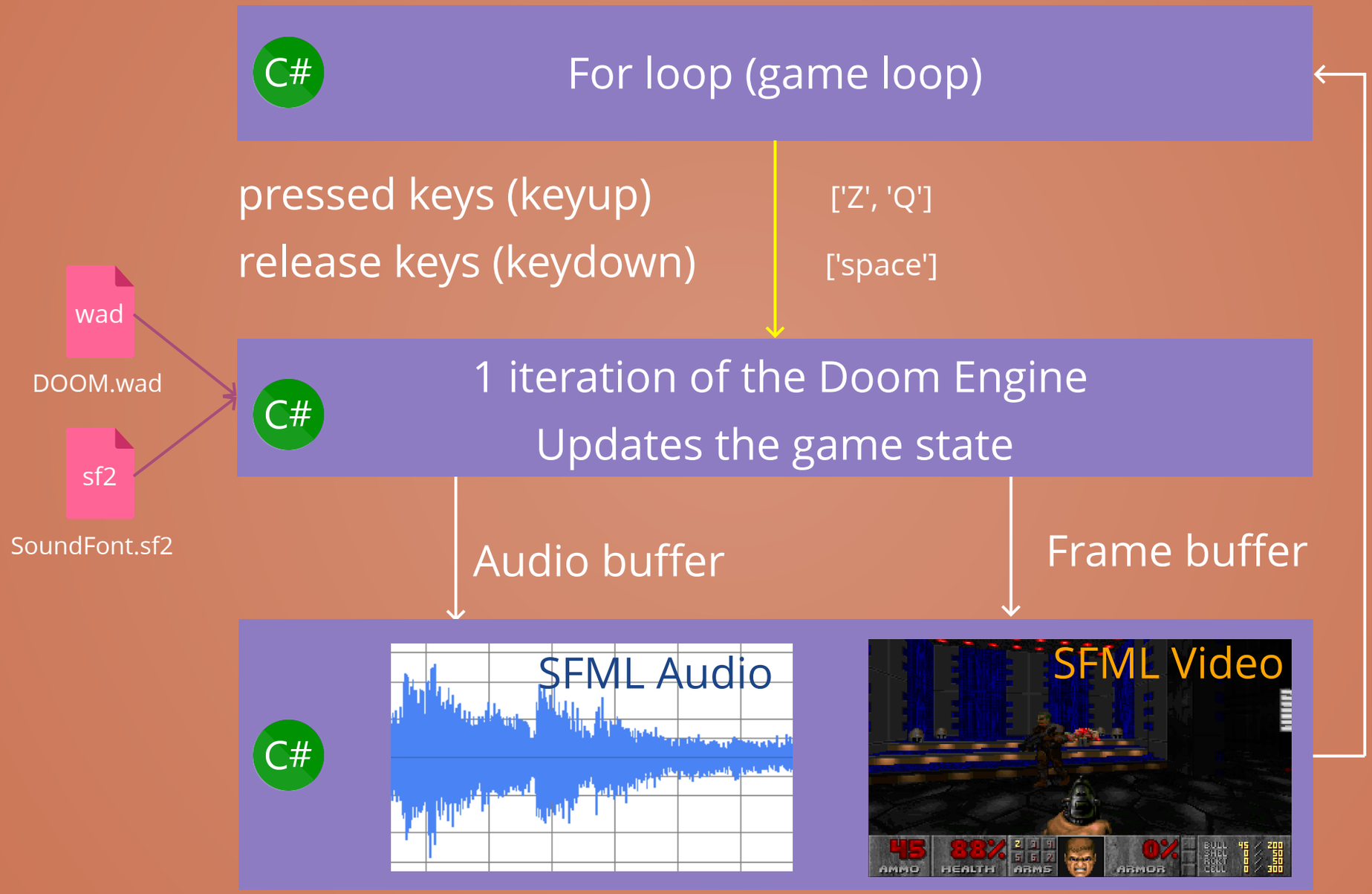
Audio buffer

Frame buffer

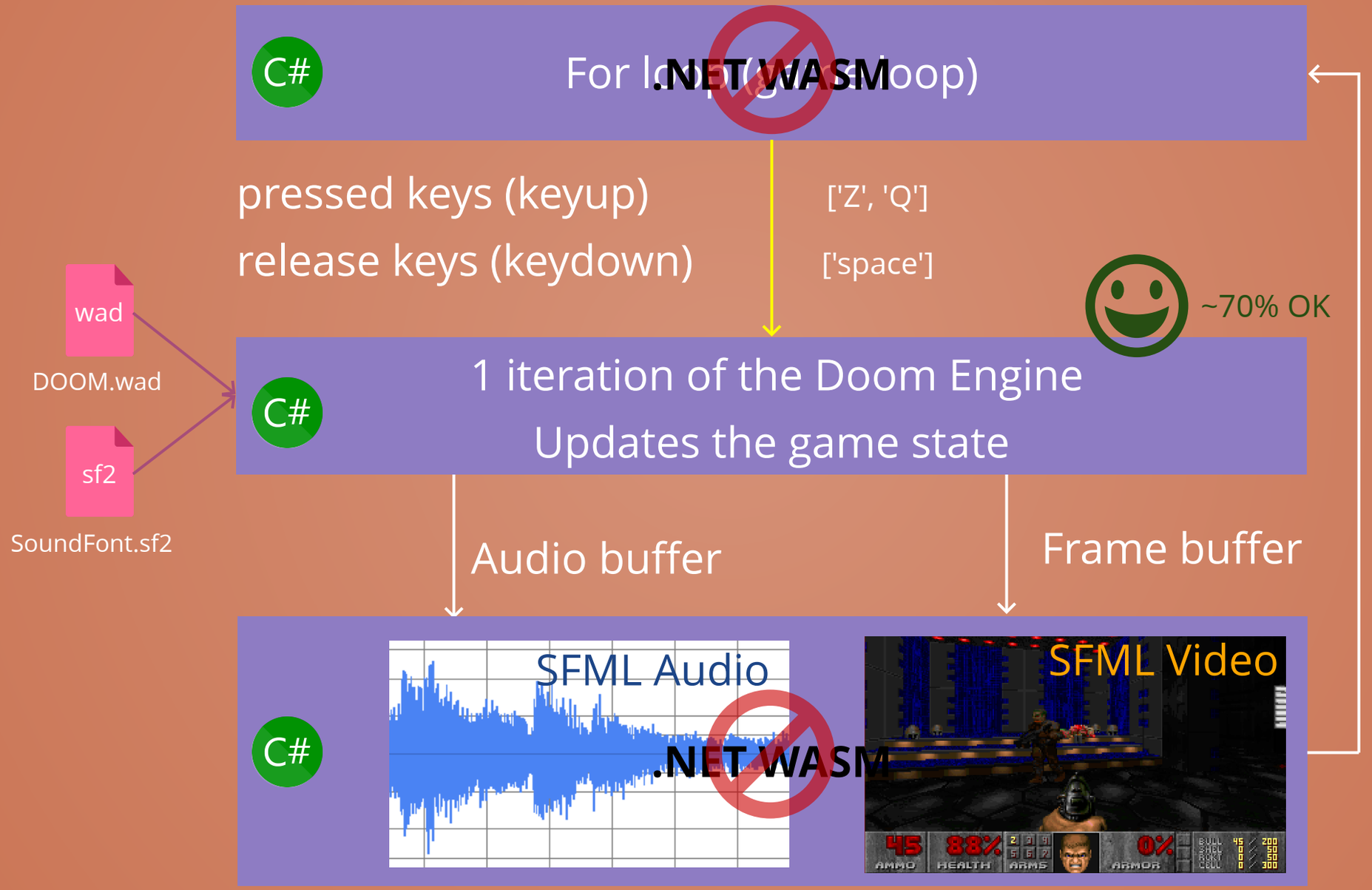
C#



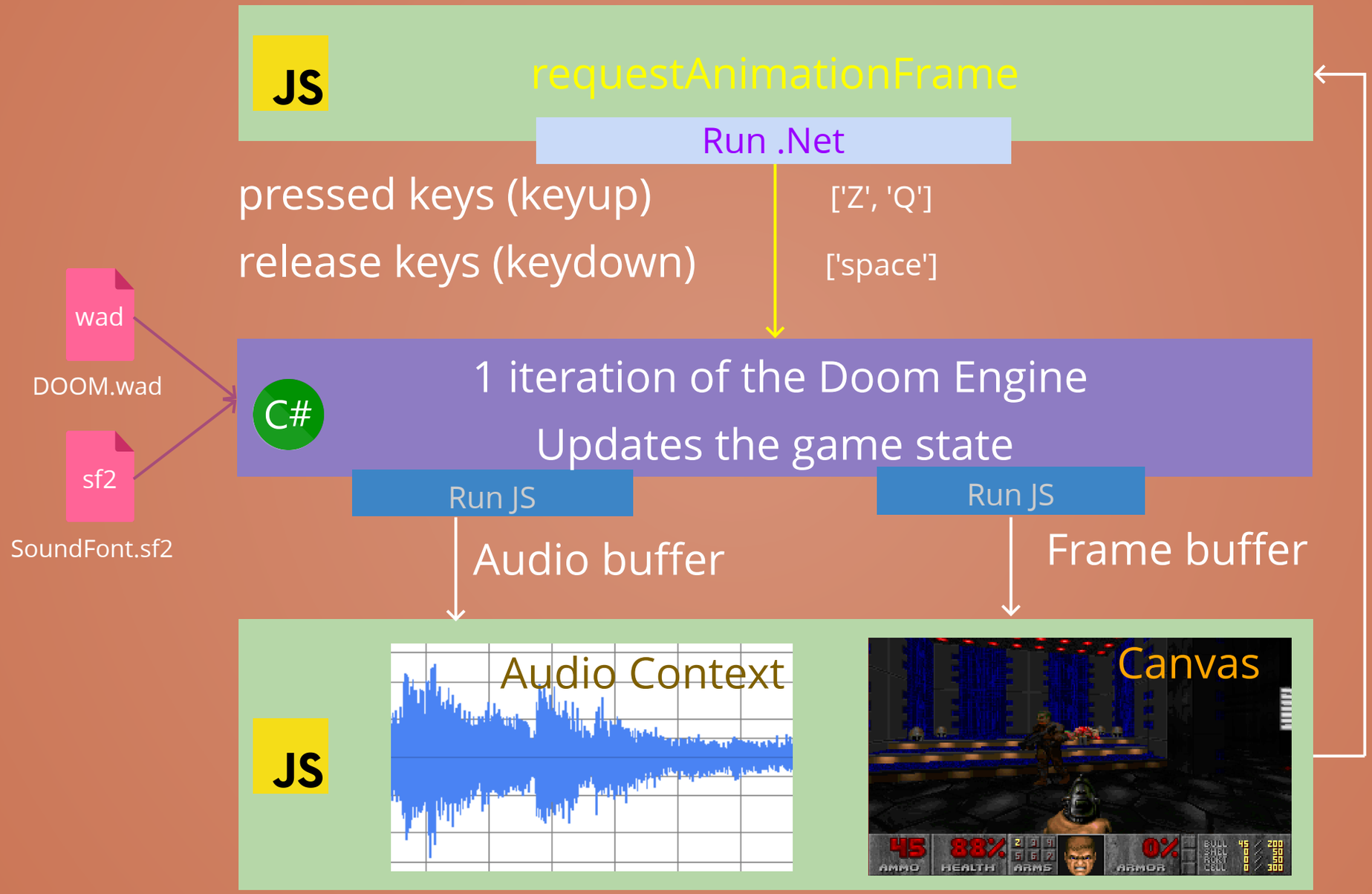
ManagedDoom V1 architecture



ManagedDoom V1 architecture

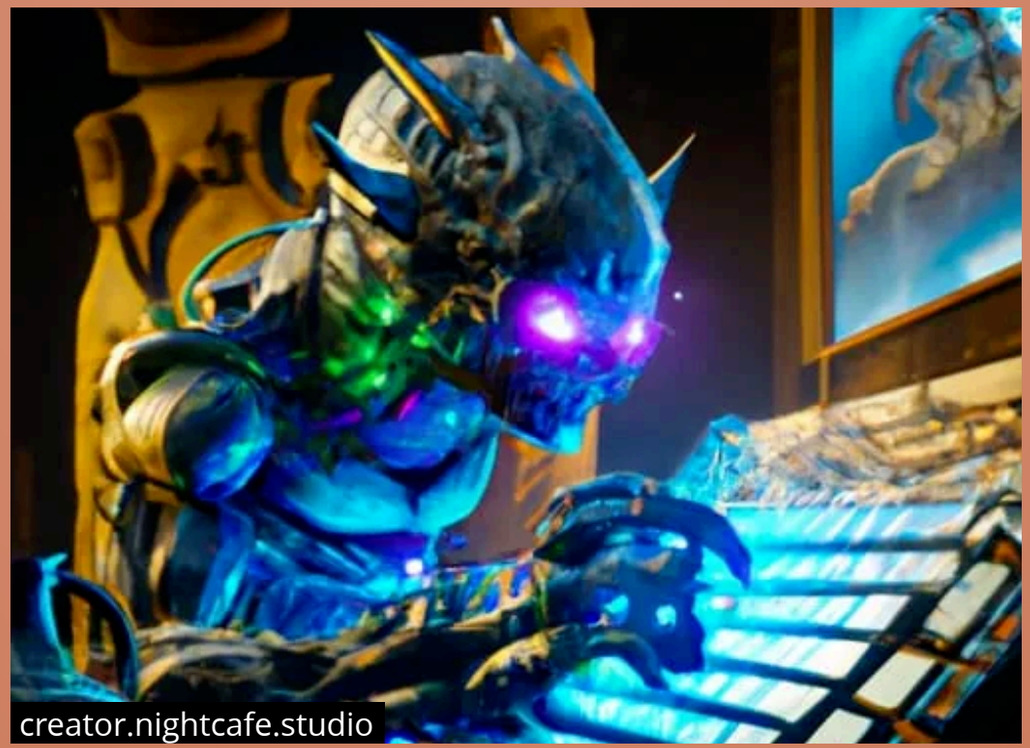


Blazor Doom architecture



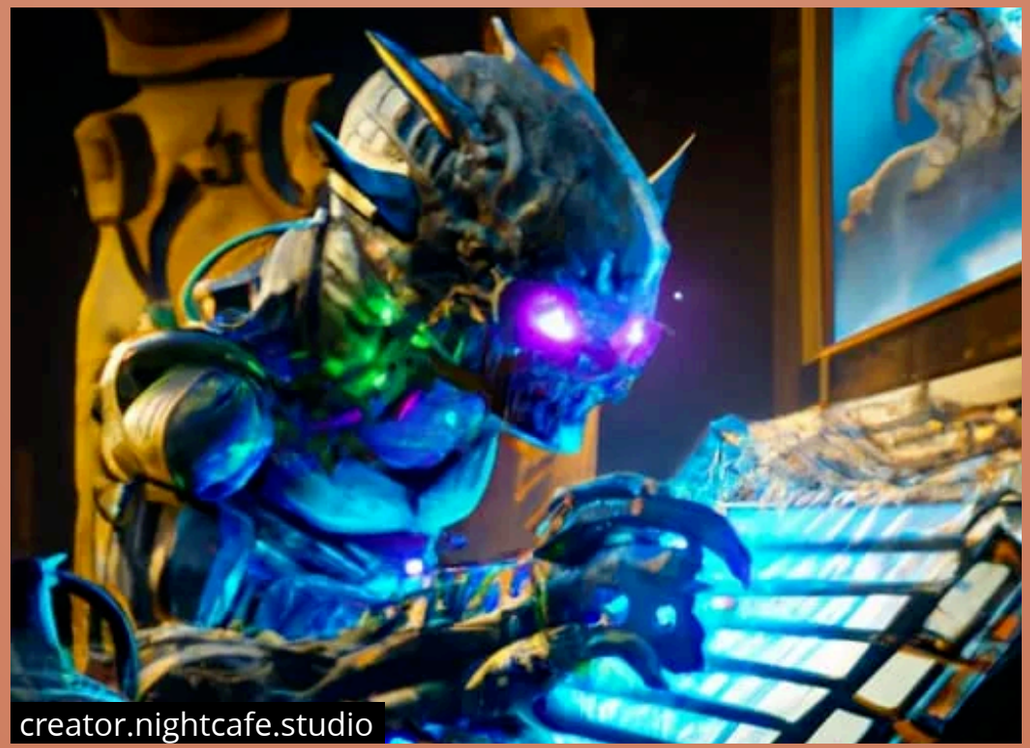
Porting strategy

- Clone **sinshu/managed-doom** and compile to **.NET WASM**
- Reverse engineer SFML:



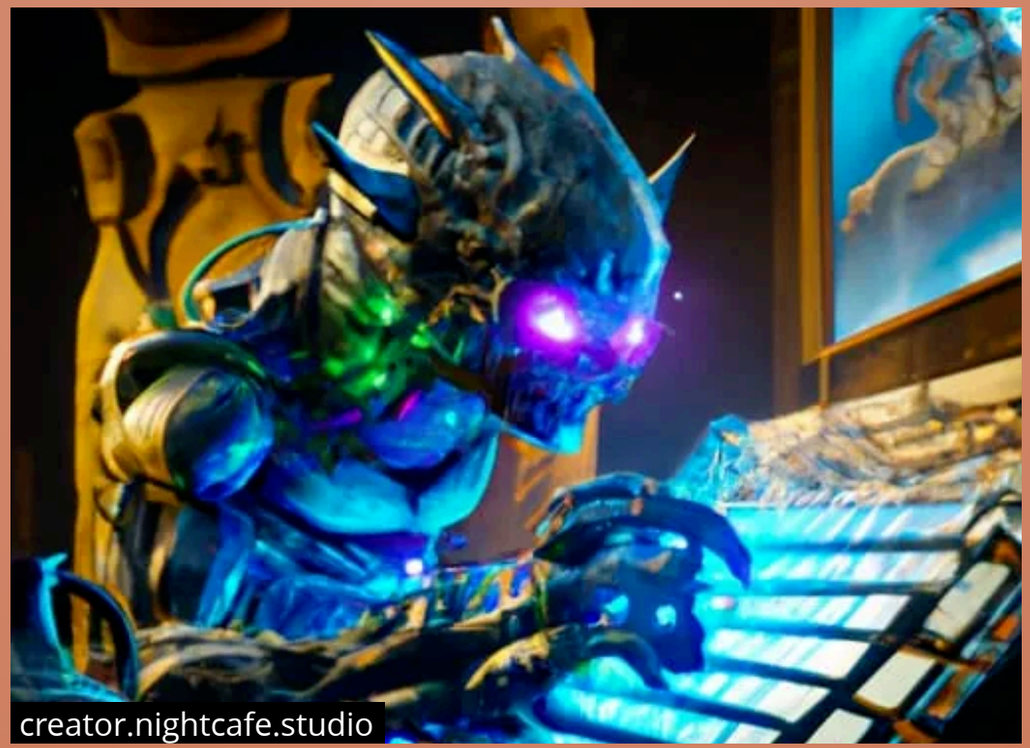
Porting strategy

- Clone **sinshu/managed-doom** and compile to **.NET WASM**
- Reverse engineer SFML:
  - Put empty implementations where it fails and matk them (*"TODO: implement"*)



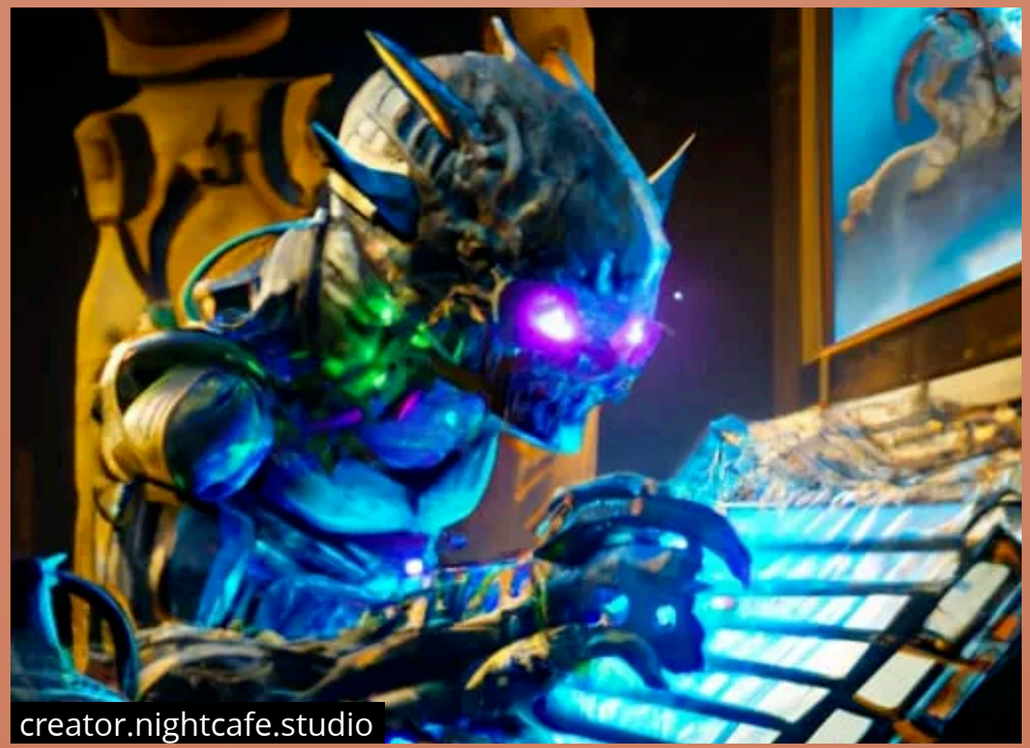
Porting strategy

- Clone **sinshu/managed-doom** and compile to **.NET WASM**
- Reverse engineer SFML:
  - Put empty implementations where it fails and matk them (*"TODO: implement"*)
  - Implement little by little using guesswork and logic



Porting strategy

- Clone **sinshu/managed-doom** and compile to **.NET WASM**
- Reverse engineer SFML:
  - Put empty implementations where it fails and matk them (*"TODO: implement"*)
  - Implement little by little using guesswork and logic
  - Delegate to JS when necessary



Example: SFML.Audio.SoundBuffer is not available

```
1 using SFML.Audio; // Not available
2
3 namespace ManagedDoom.Audio
4 {
5     public sealed class SfmlSound : ISound, IDisposable
6     {
7         private SoundBuffer[] buffers;
8     }
9 }
```

So, let's reverse-engineer it

```
 1 namespace SFML.Audio
 2 {
 3     public class SoundBuffer
 4     {
 5         public SoundBuffer(short[] samples, int v, uint sampleRate)
 6         {
 7             // TODO: implement
 8         }
 9
10         internal void Dispose()
11         {
12             // TODO: implement
13         }
14     }
15 }
```

```
namespace SFML.Audio
{
    public class SoundBuffer
    {
        public short[] samples;
        private int v;
        public uint sampleRate;

        public SoundBuffer(short[] samples, int v, uint sampleRate)
        {
            this.samples = samples;
            this.v = v;
            this.sampleRate = sampleRate;
        }

        public Time Duration { get; internal set; }

        internal void Dispose()
        {
            // TODO: implement
        }
    }
}
```

Chapter 4

Knee-deep in the code

Chapter 4

Knee-deep in the code

Entry point

```
 1 <html>
 2   <head>
 3     <!-- Start the game loop -->
 4     <script type="module" src="./main.js"></script>
 5     <!-- Load the .net runtime and our compiled C# code ! -->
 6     <script type="module" src="./dotnet.js"></script>
 7     <link rel="prefetch" href="./dotnet.wasm"/>
 8    </head>
 9   <body>
10     <canvas id="canvas" width="320" height="200"
11             style="image-rendering: pixelated" />
12   </body>
13 </html>
```

Entry point

```
 1 <html>
 2   <head>
 3     <!-- Start the game loop -->
 4     <script type="module" src="./main.js"></script>
 5     <!-- Load the .net runtime and our compiled C# code ! -->
 6     <script type="module" src="./dotnet.js"></script>
 7     <link rel="prefetch" href="./dotnet.wasm"/>
 8    </head>
 9   <body>
10     <canvas id="canvas" width="320" height="200"
11             style="image-rendering: pixelated" />
12   </body>
13 </html>
```

Entry point

```
1  <html>
2    <head>
3      <!-- Start the game loop -->
4      <script type="module" src="./main.js"></script>
5      <!-- Load the .net runtime and our compiled C# code ! -->
6      <script type="module" src="./dotnet.js"></script>
7      <link rel="prefetch" href="./dotnet.wasm"/>
8     </head>
9    <body>
10     <canvas id="canvas" width="320" height="200"
11             style="image-rendering: pixelated" />
12   </body>
13 </html>
```

Entry point

```
<html>
  <head>
    <!-- Start the game loop -->
    <script type="module" src="./main.js"></script>
    <!-- Load the .net runtime and our compiled C# code ! -->
    <script type="module" src="./dotnet.js"></script>
    <link rel="prefetch" href="./dotnet.wasm"/>
   </head>
  <body>
    <canvas id="canvas" width="320" height="200"
            style="image-rendering: pixelated" />
  </body>
</html>
```

Running the gameloop

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Running the gameloop

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Running the gameloop

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Running the gameloop

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Running the gameloop

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Running the gameloop

main.js

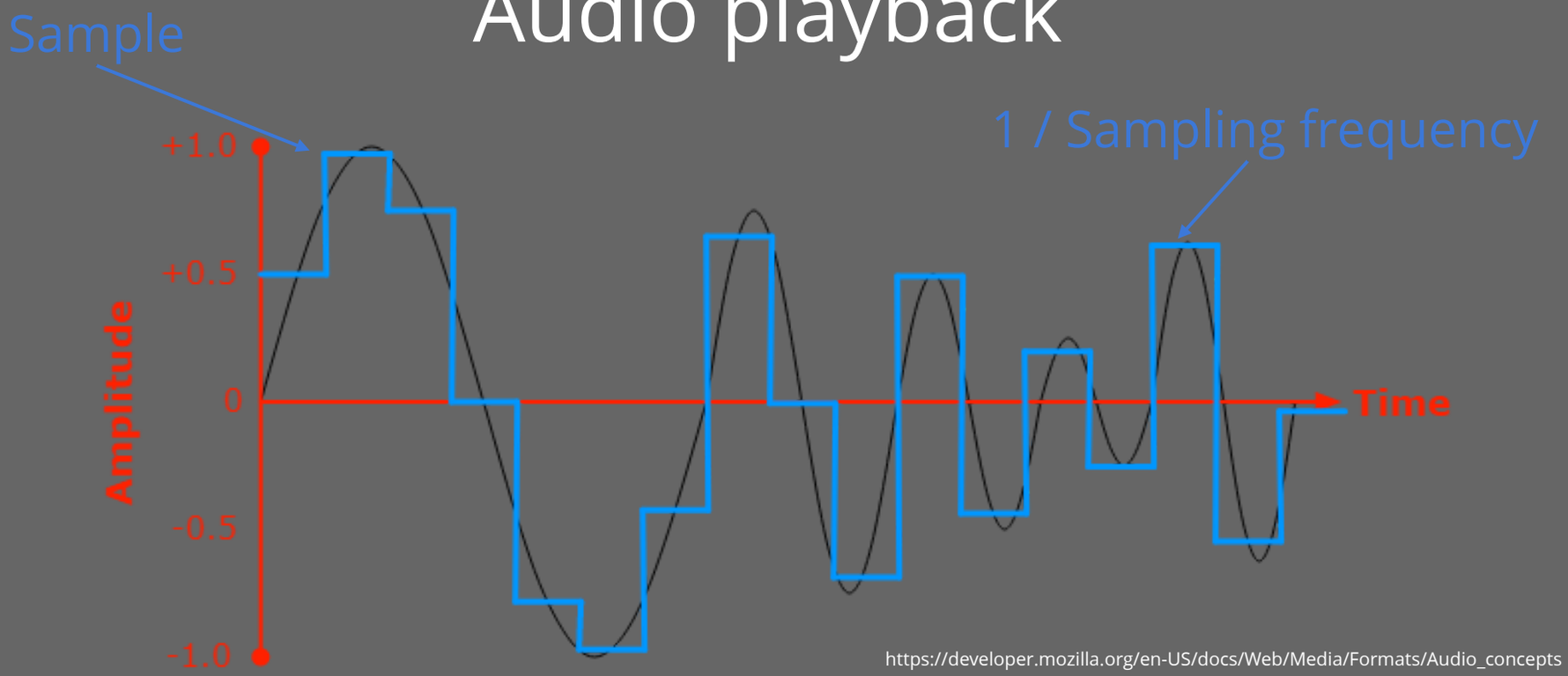
```js
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

JS

```cs
public partial class MainJS // this name is required
{
  public static void Main()
  {
    app = new ManagedDoom.DoomApplication();
  }

  [JSExport] // Can be imported from JS
  public static void UpdateGameState(int[] keys)
  { // computes the next frame and sounds
     managedDoom.UpdateGameState(keys);
  }
}
```

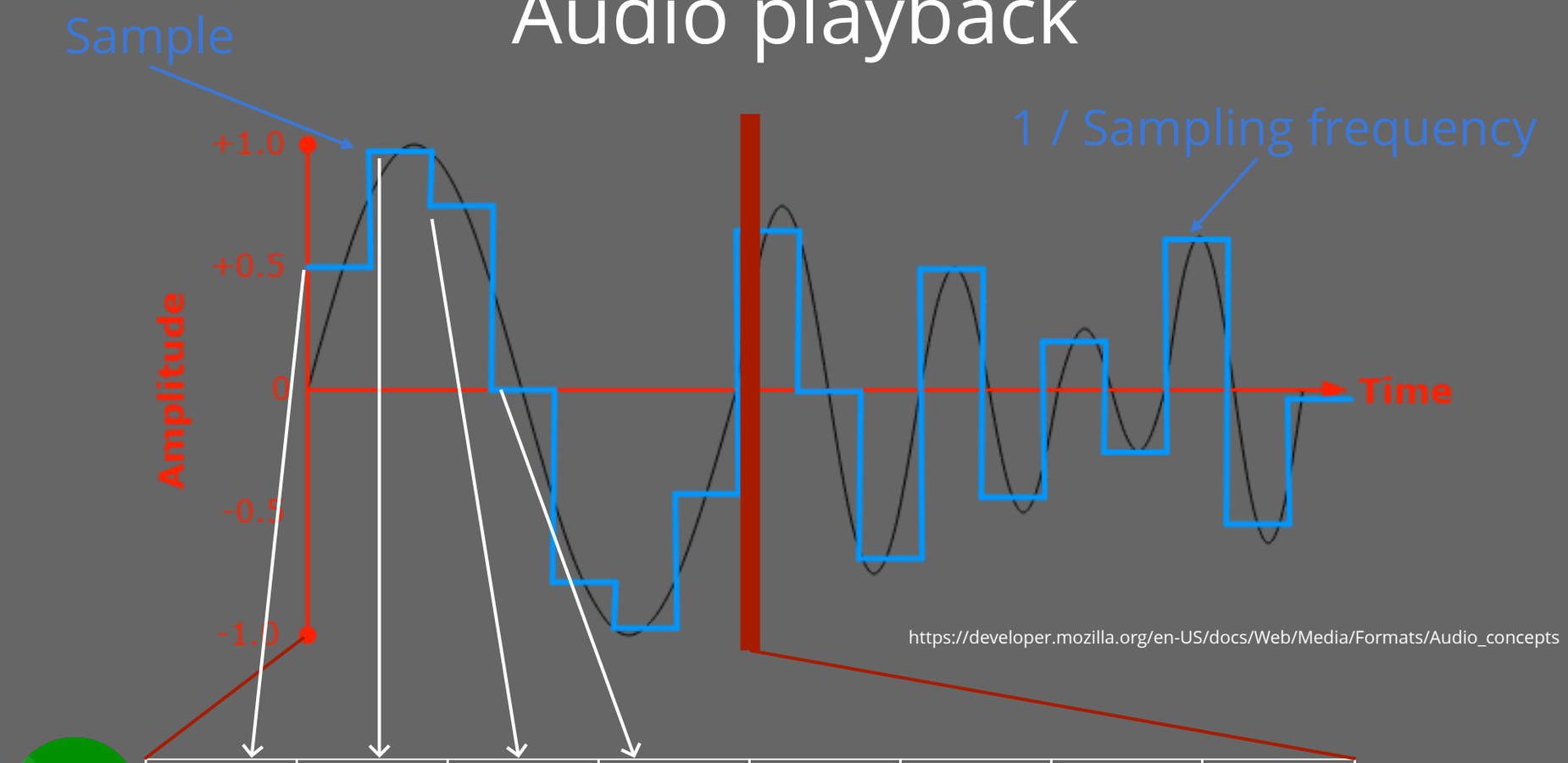
C#

Audio playback



https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

Audio playback

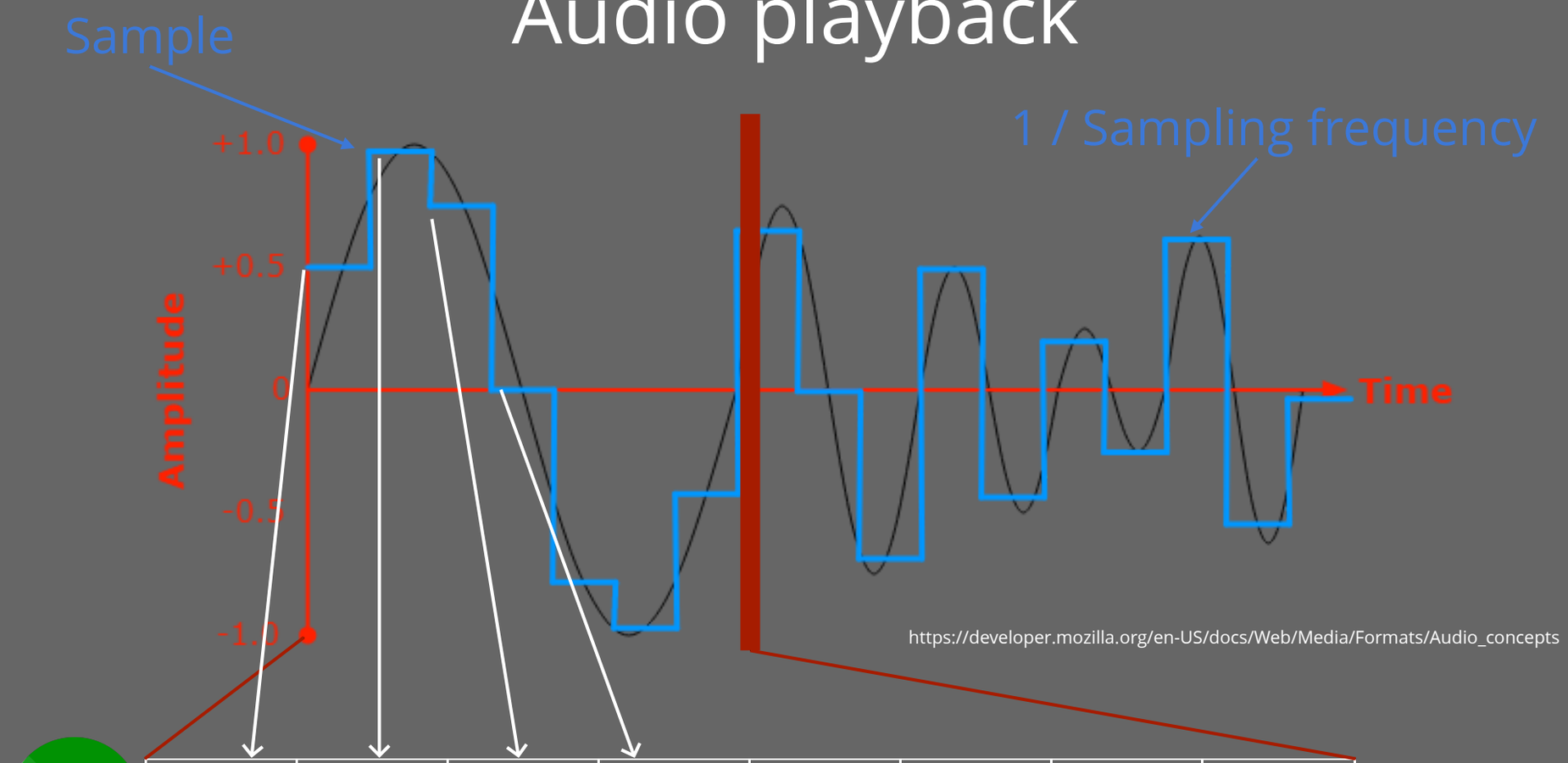


C#

0.5	1	0.75	0	-0.75	-1	-0.5	0

+ Sampling frequency

Audio playback



C#

0.5	1	0.75	0	-0.75	-1	-0.5	0

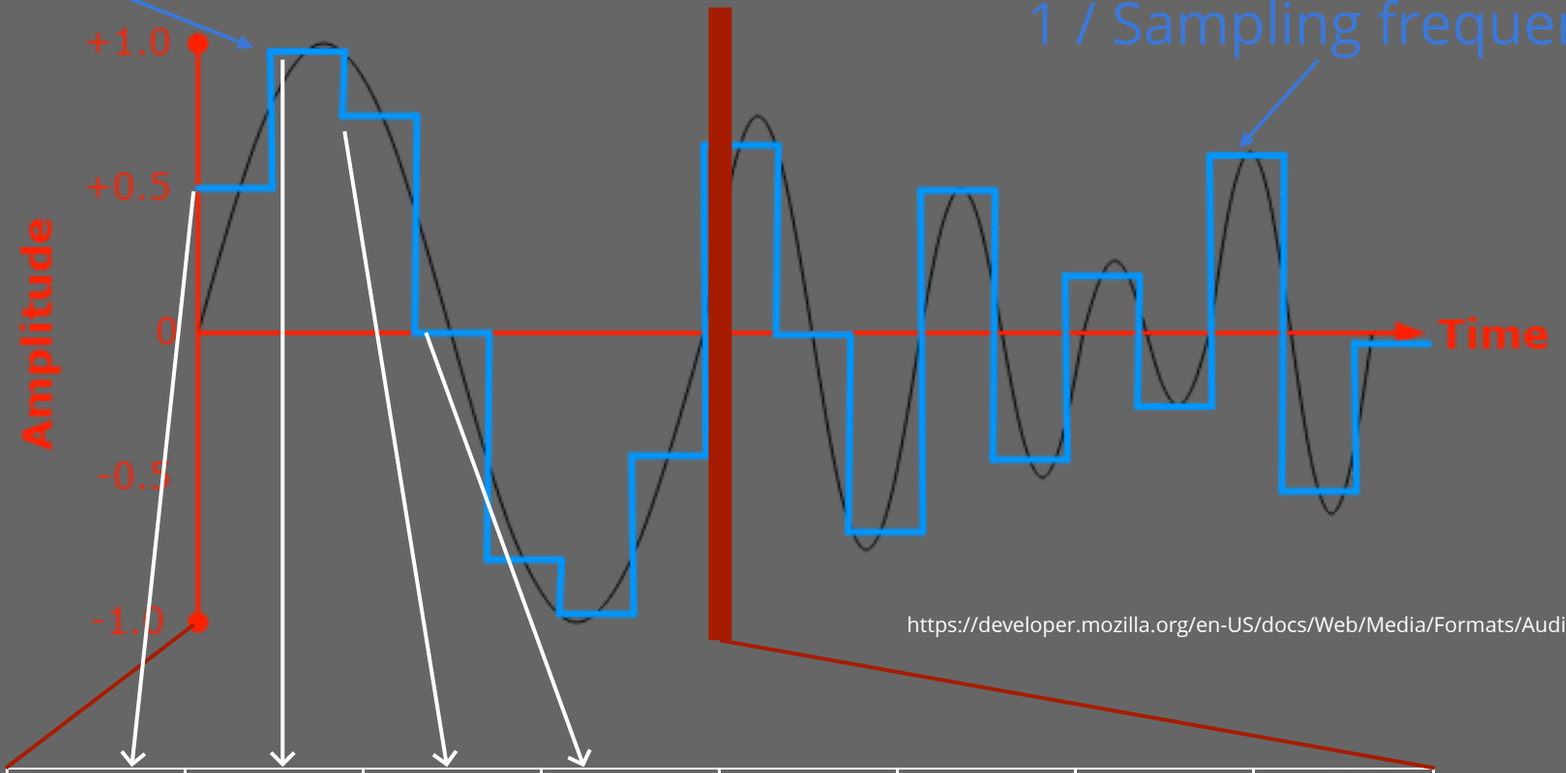
+ Sampling frequency

JS AudioContext

Audio playback

Sample

1 / Sampling frequency



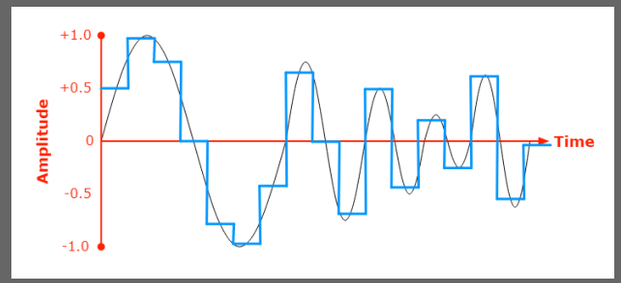
https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

C#

0.5	1	0.75	0	-0.75	-1	-0.5	0

+ Sampling frequency

JS AudioContext



Audio playback

```
1  void PlayCurrentFrameSound(SoundBuffer soundBuffer)
2  {
3      int[] samples = Array.ConvertAll(soundBuffer.samples, Convert.ToInt32);
4      BlazorDoom.Renderer.playSoundOnJS(samples, (int)soundBuffer.sampleRate);
5  }
```

C#

```
 1  namespace BlazorDoom
 2  {
 3      [SupportedOSPlatform("browser")]
 4      public partial class Renderer
 5      {
 6          [JSImport("playSound", "blazorDoom/renderer.js")]
 7          internal static partial string playSoundOnJS(
 8              int[] samples,
 9              int sampleRate
10          );
11      }
12  }
```

C#

Audio playback

```
export function playSound(samples, sampleRate) {
    audioContext = new AudioContext({
        sampleRate: sampleRate,
    });
    const length = samples.length;
    const audioBuffer = audioContext.createBuffer(
        1,
        length,
        sampleRate
    );

    var channelData = audioBuffer.getChannelData(0);
    for (let i = 0; i < length; i++) {
        // noralize the sample to be between -1 and 1
        channelData[i] = samples[i] / 0xffff;
    }

    var source = audioContext.createBufferSource();
    source.buffer = audioBuffer;
    source.connect(audioContext.destination);
    source.start();
}
```

JS

Audio playback

```
export function playSound(samples, sampleRate) {
    audioContext = new AudioContext({
        sampleRate: sampleRate,
    });
    const length = samples.length;
    const audioBuffer = audioContext.createBuffer(
        1,
        length,
        sampleRate
    );

    var channelData = audioBuffer.getChannelData(0);
    for (let i = 0; i < length; i++) {
        // noralize the sample to be between -1 and 1
        channelData[i] = samples[i] / 0xffff;
    }

    var source = audioContext.createBufferSource();
    source.buffer = audioBuffer;
    source.connect(audioContext.destination);
    source.start();
}
```

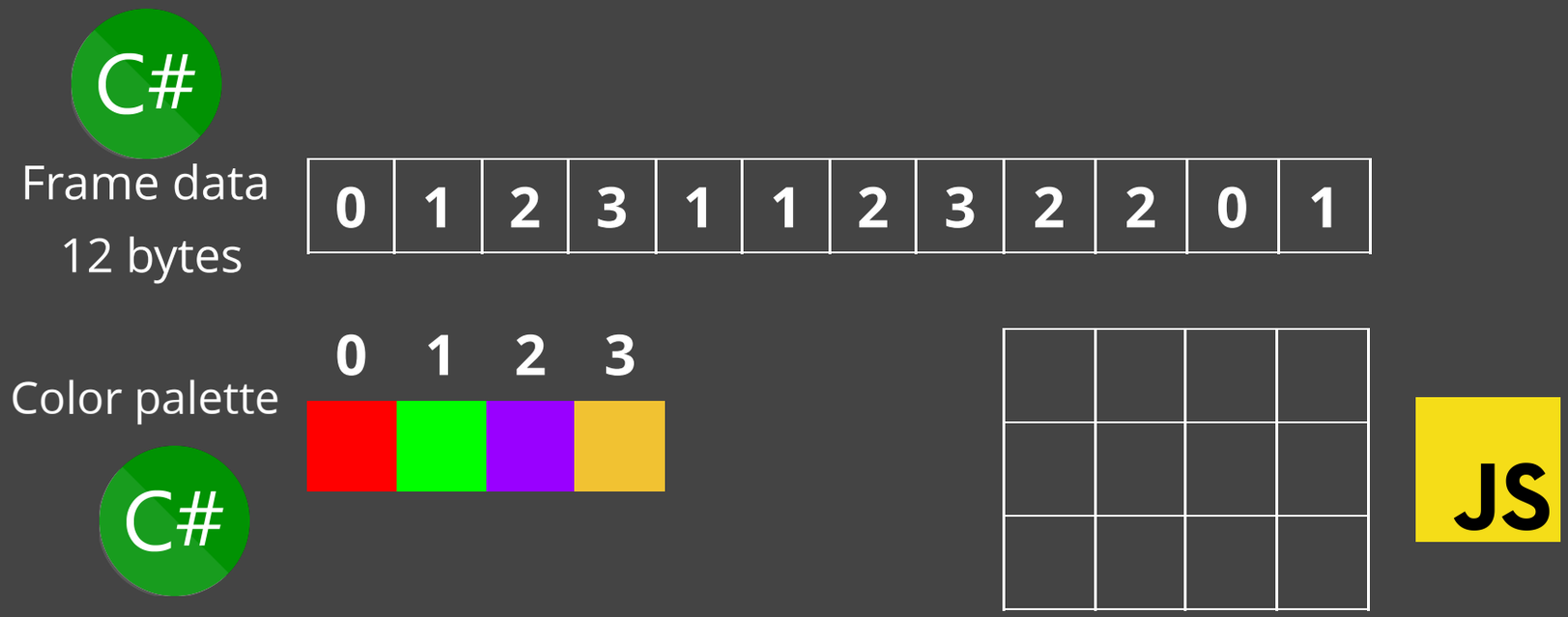
JS

Audio playback

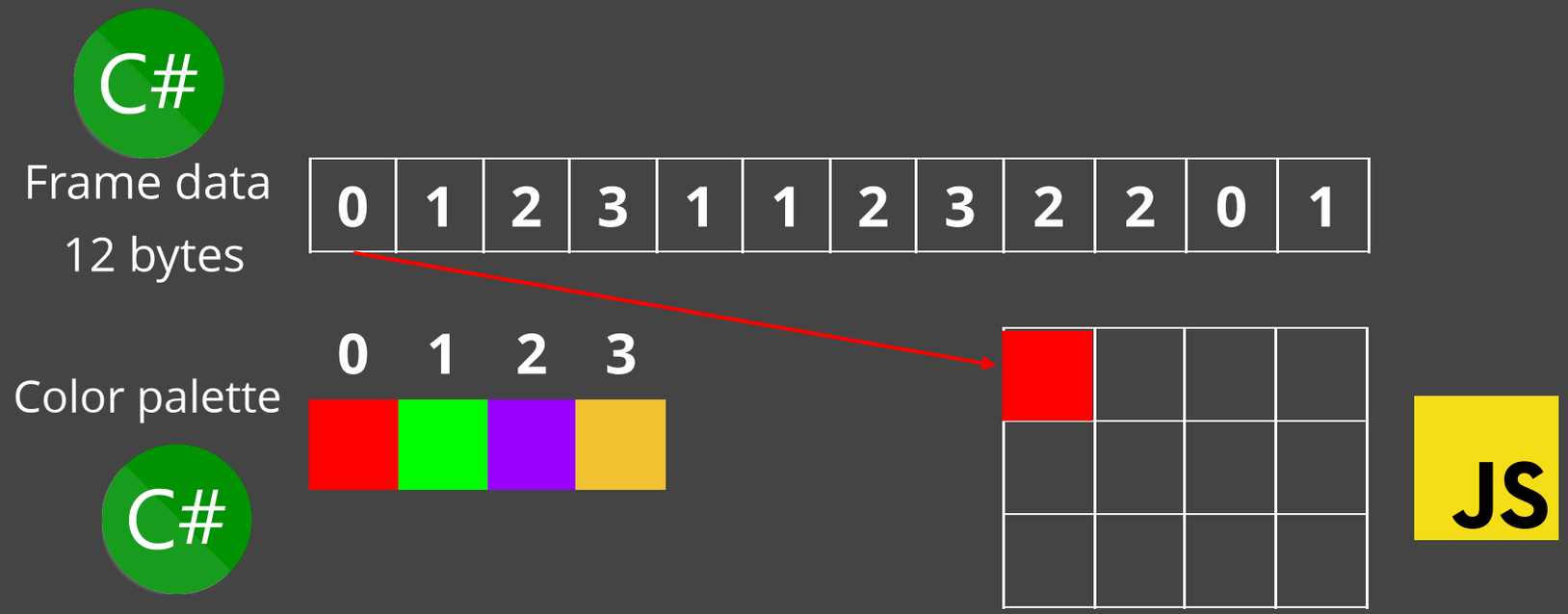
```
export function playSound(samples, sampleRate) {
    audioContext = new AudioContext({
        sampleRate: sampleRate,
    });
    const length = samples.length;
    const audioBuffer = audioContext.createBuffer(
        1,
        length,
        sampleRate
    );

    var channelData = audioBuffer.getChannelData(0);
    for (let i = 0; i < length; i++) {
        // noralize the sample to be between -1 and 1
        channelData[i] = samples[i] / 0xffff;
    }

    var source = audioContext.createBufferSource();
    source.buffer = audioBuffer;
    source.connect(audioContext.destination);
    source.start();
}
```

JS

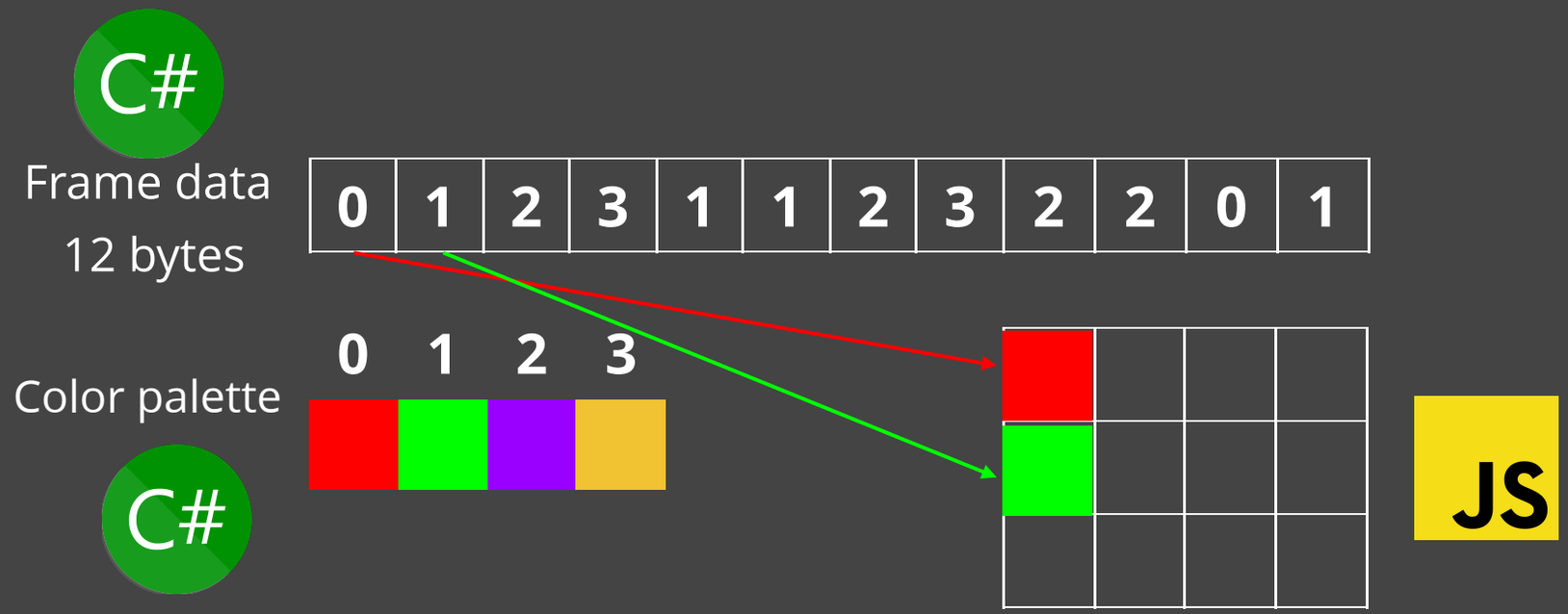
From 1D frame to a 2D frame



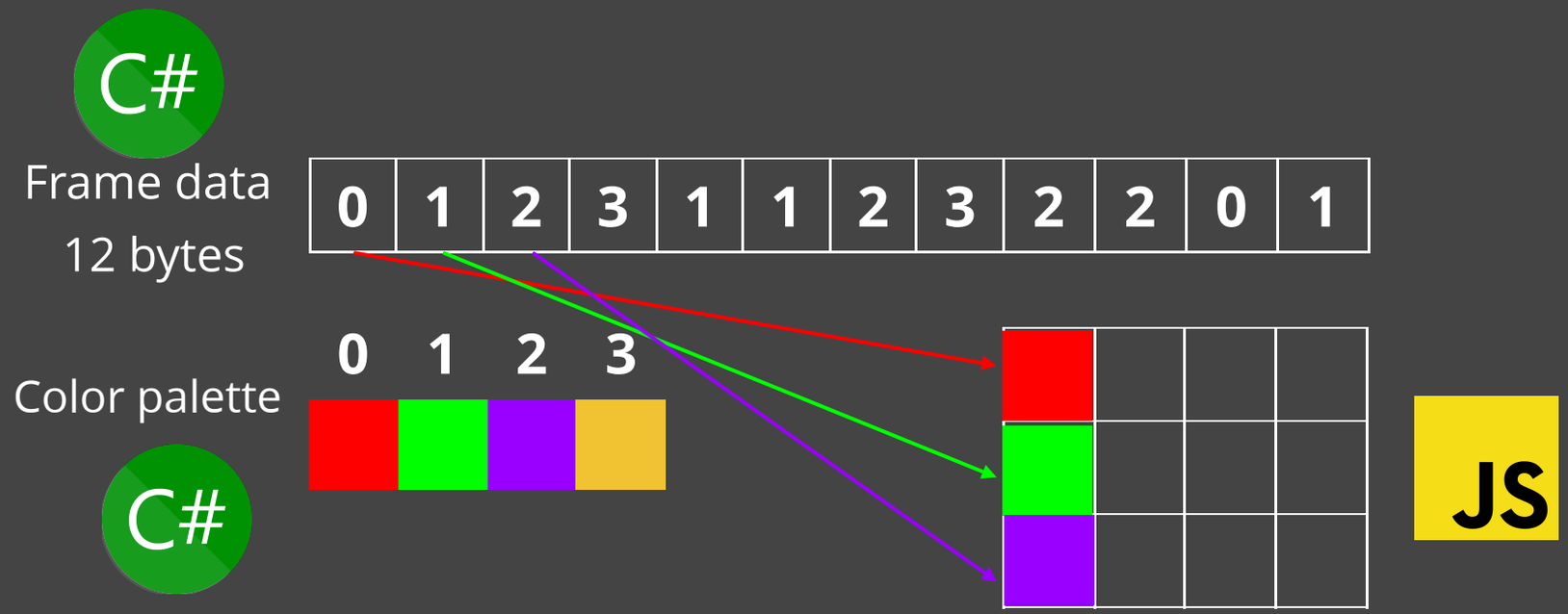
From 1D frame to a 2D frame



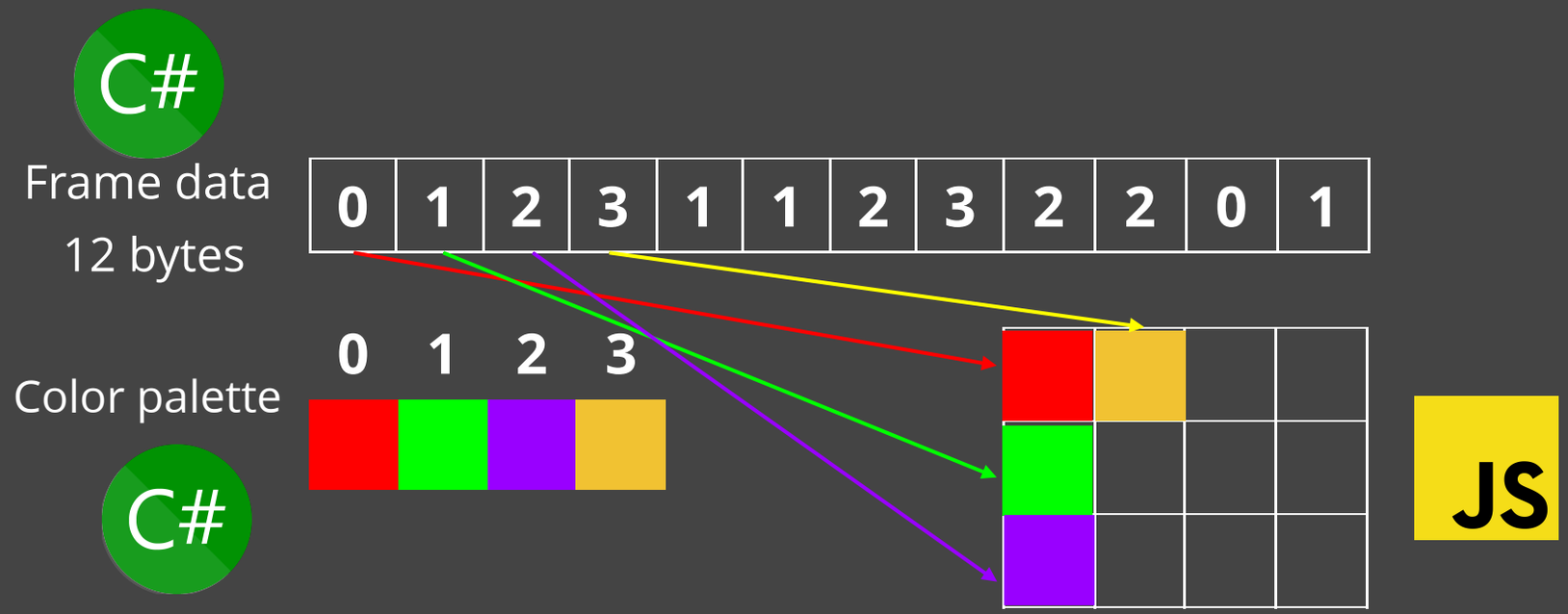
From 1D frame to a 2D frame



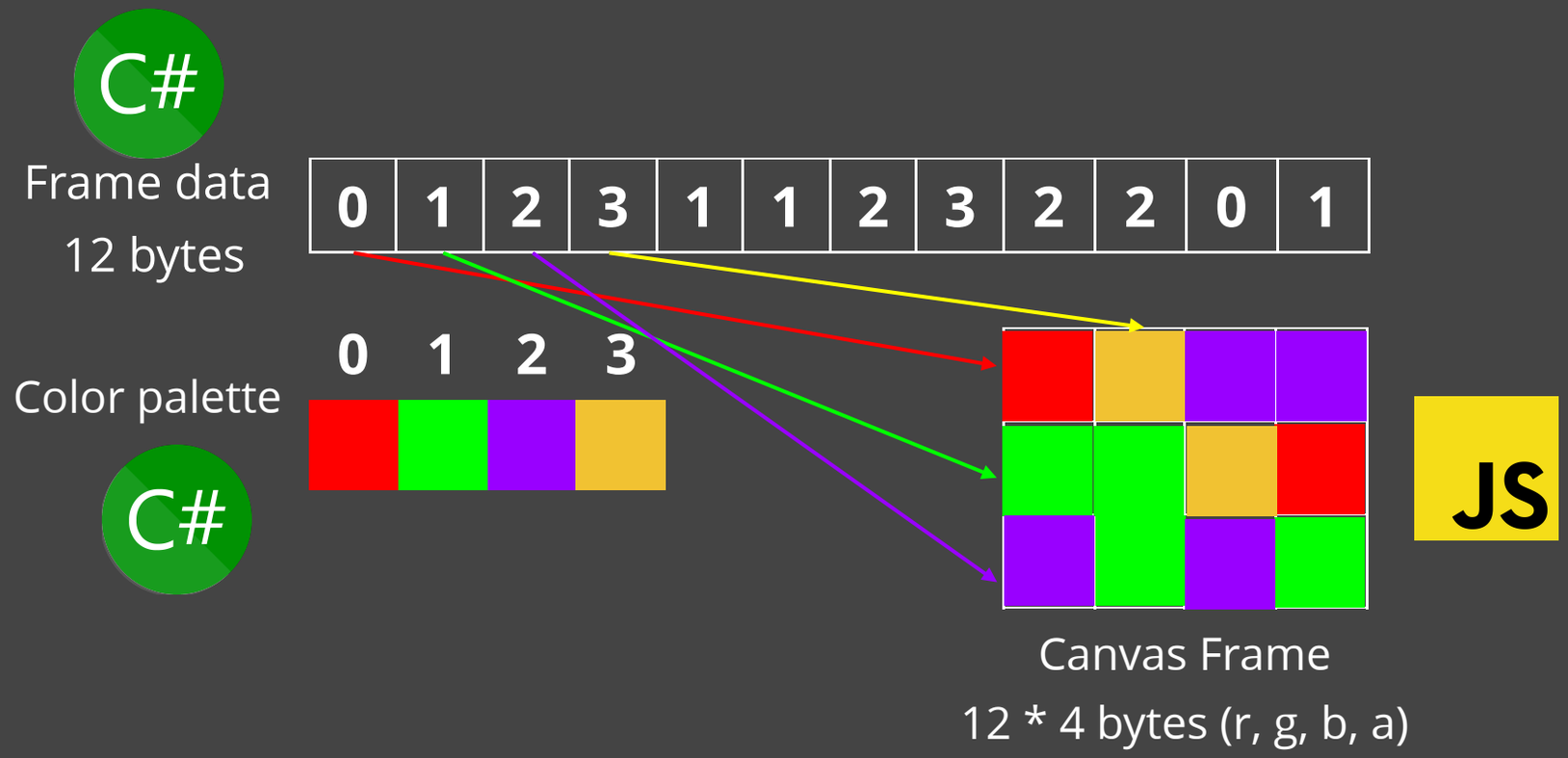
From 1D frame to a 2D frame



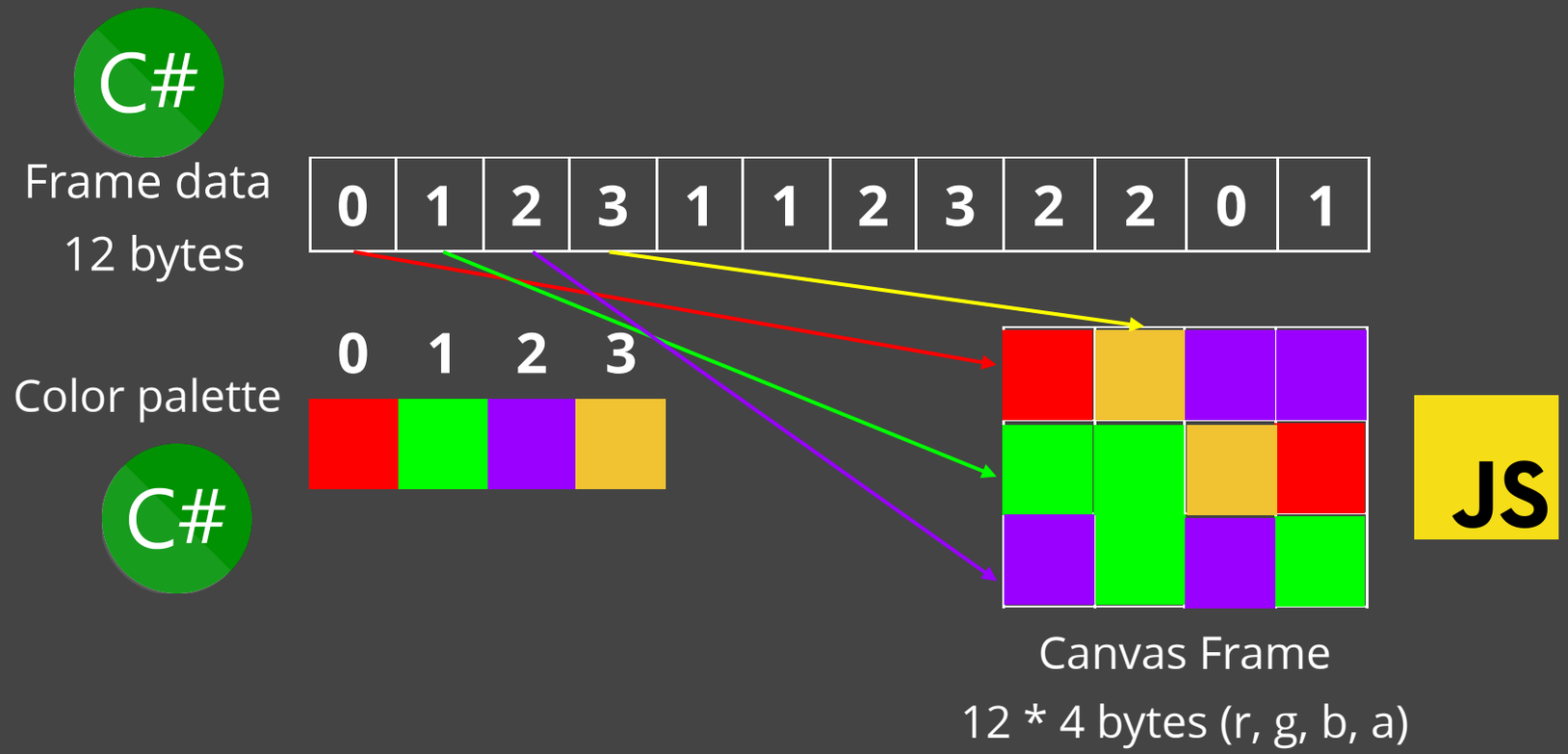
From 1D frame to a 2D frame



From 1D frame to a 2D frame



From 1D frame to a 2D frame



- Doom uses color indexing
- Image built from top to bottom and from left to right

Frame rendering: from C# to JS

```
1 public class DoomRenderer
2 {
3   // Called by updateGameState
4   private void Display(uint[] colors)
5   {
6     BlazorDoom.Renderer.renderOnJS(screen.Data, (int[])((object)colors));
7   }
8 }
```

C#

```
 1 namespace BlazorDoom
 2 {
 3     [SupportedOSPlatform("browser")]
 4     public partial class Renderer
 5     {
 6         [JSImport("drawOnCanvas", "blazorDoom/renderer.js")]
 7         internal static partial string renderOnJS(byte[] screenData,
 8                                                   int[] colors);
 9     }
10 }
```

C#

```
1 export function drawOnCanvas(screenData, colors) {
2   //
3 }
```

JS

Rendering a Frame buffer

```
export function drawOnCanvas(screenData, colors) {
    const context = getCanvas().context;
    const imageData = context.createImageData(320, 200);
    let y = 0;
    let x = 0;
    for (let i = 0; i < screenData.length; i += 1) {
        const dataIndex = (y * width + x) * 4;
        setSinglePixel(imageData, dataIndex, colors, screenData[i]);
        if (y >= height - 1) {
            y = 0;
            x += 1;
        } else {
            y += 1;
        }
    }
    context.putImageData(imageData, 0, 0);
}

function setSinglePixel(imageData, dataIndex, colors, colorIndex) {
    const color = colors[colorIndex];
    imageData.data[dataIndex] = color & 0xff; // R
    imageData.data[dataIndex + 1] = (color >> 8) & 0xff; // G
    imageData.data[dataIndex + 2] = (color >> 16) & 0xff; // B
    imageData.data[dataIndex + 3] = 255; // Alpha
}
```

JS

Rendering a Frame buffer

```
export function drawOnCanvas(screenData, colors) {
    const context = getCanvas().context;
    const imageData = context.createImageData(320, 200);
    let y = 0;
    let x = 0;
    for (let i = 0; i < screenData.length; i += 1) {
        const dataIndex = (y * width + x) * 4;
        setSinglePixel(imageData, dataIndex, colors, screenData[i]);
        if (y >= height - 1) {
            y = 0;
            x += 1;
        } else {
            y += 1;
        }
    }
    context.putImageData(imageData, 0, 0);
}

function setSinglePixel(imageData, dataIndex, colors, colorIndex) {
    const color = colors[colorIndex];
    imageData.data[dataIndex] = color & 0xff; // R
    imageData.data[dataIndex + 1] = (color >> 8) & 0xff; // G
    imageData.data[dataIndex + 2] = (color >> 16) & 0xff; // B
    imageData.data[dataIndex + 3] = 255; // Alpha
}
```

JS

Rendering a Frame buffer

```
1  export function drawOnCanvas(screenData, colors) {
2      const context = getCanvas().context;
3      const imageData = context.createImageData(320, 200);
4      let y = 0;
5      let x = 0;
6      for (let i = 0; i < screenData.length; i += 1) {
7          const dataIndex = (y * width + x) * 4;
8          setSinglePixel(imageData, dataIndex, colors, screenData[i]);
9          if (y >= height - 1) {
10             y = 0;
11             x += 1;
12         } else {
13             y += 1;
14         }
15     }
16     context.putImageData(imageData, 0, 0);
17 }
18
19 function setSinglePixel(imageData, dataIndex, colors, colorIndex) {
20     const color = colors[colorIndex];
21     imageData.data[dataIndex] = color & 0xff; // R
22     imageData.data[dataIndex + 1] = (color >> 8) & 0xff; // G
23     imageData.data[dataIndex + 2] = (color >> 16) & 0xff; // B
24     imageData.data[dataIndex + 3] = 255; // Alpha
25 }
```

JS

CHAPTER 5

DEMO(N)

EPILOGUE

NEXT STEPS AND CONCLUSION

Next steps

- Better BGM (Background Music) implementation
- Update to ManagedDoom V2 (replaced SFML with Silk.Net)
- Make this port an official part of ManagedDoom

Key takeaways

Key takeaways

- WASM possibilities are limitless

Key takeaways

- WASM possibilities are limitless
- Porting a game is great and fun way to learn programming

SLIDE DECK

SOURCE CODE

SLIDES OVER

THANKS FOR PLAYING !
ANY QUESTION ?

Yassine Benabbas

@yostane

Links

- https://mspoweruser.com/this-doom-digital-camera-source-port-is-amazing-and-bizarre/
- https://www.link-cable.com/top-10-weird-doom-ports/
- pixabay.com
- Doom text generator: https://c.eev.ee/doom-text-generator