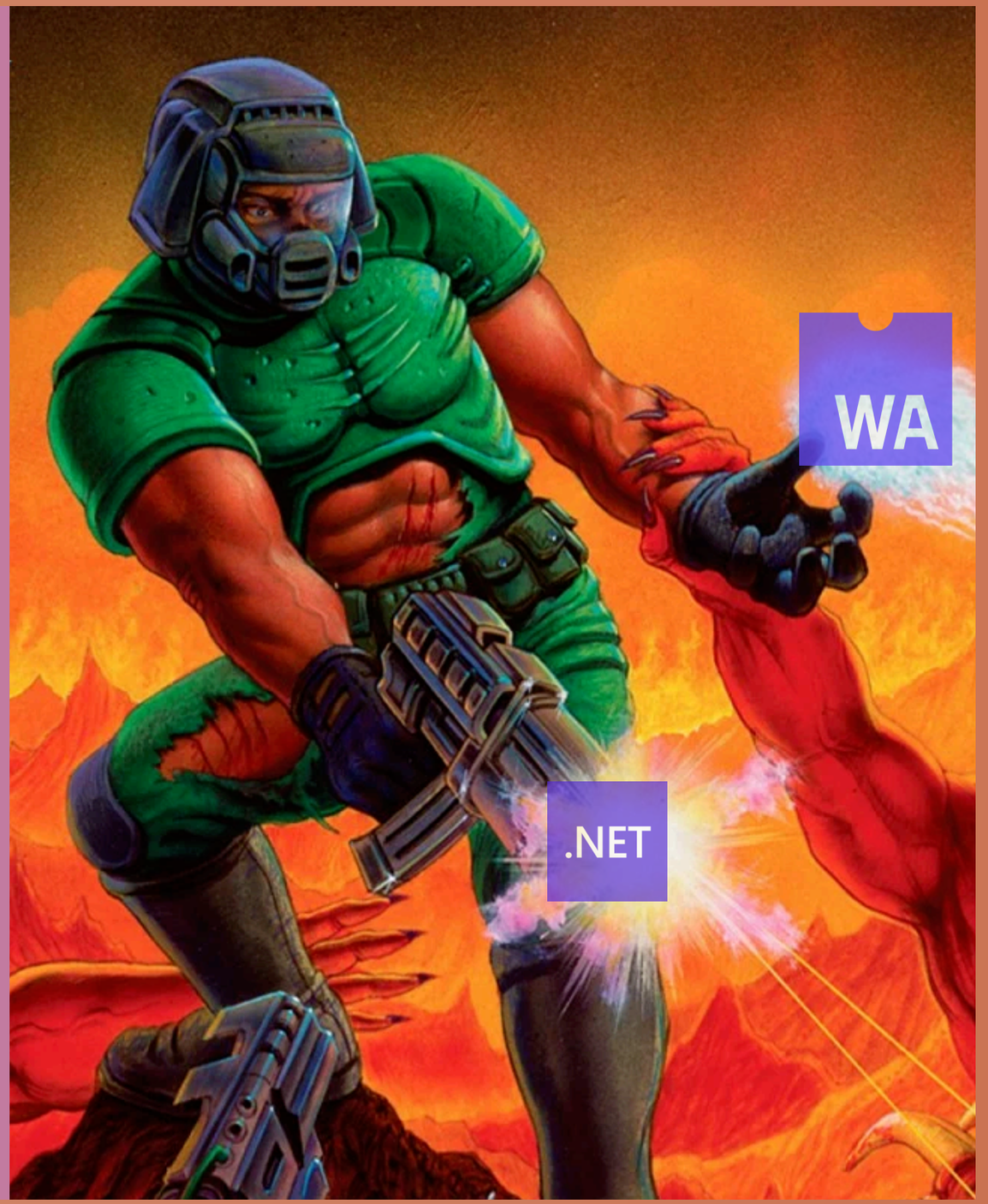
How it started

How it's going

How I Ported DOOM to the Browser with WebAssembly



FORK IT!

Yassine
Benabbas

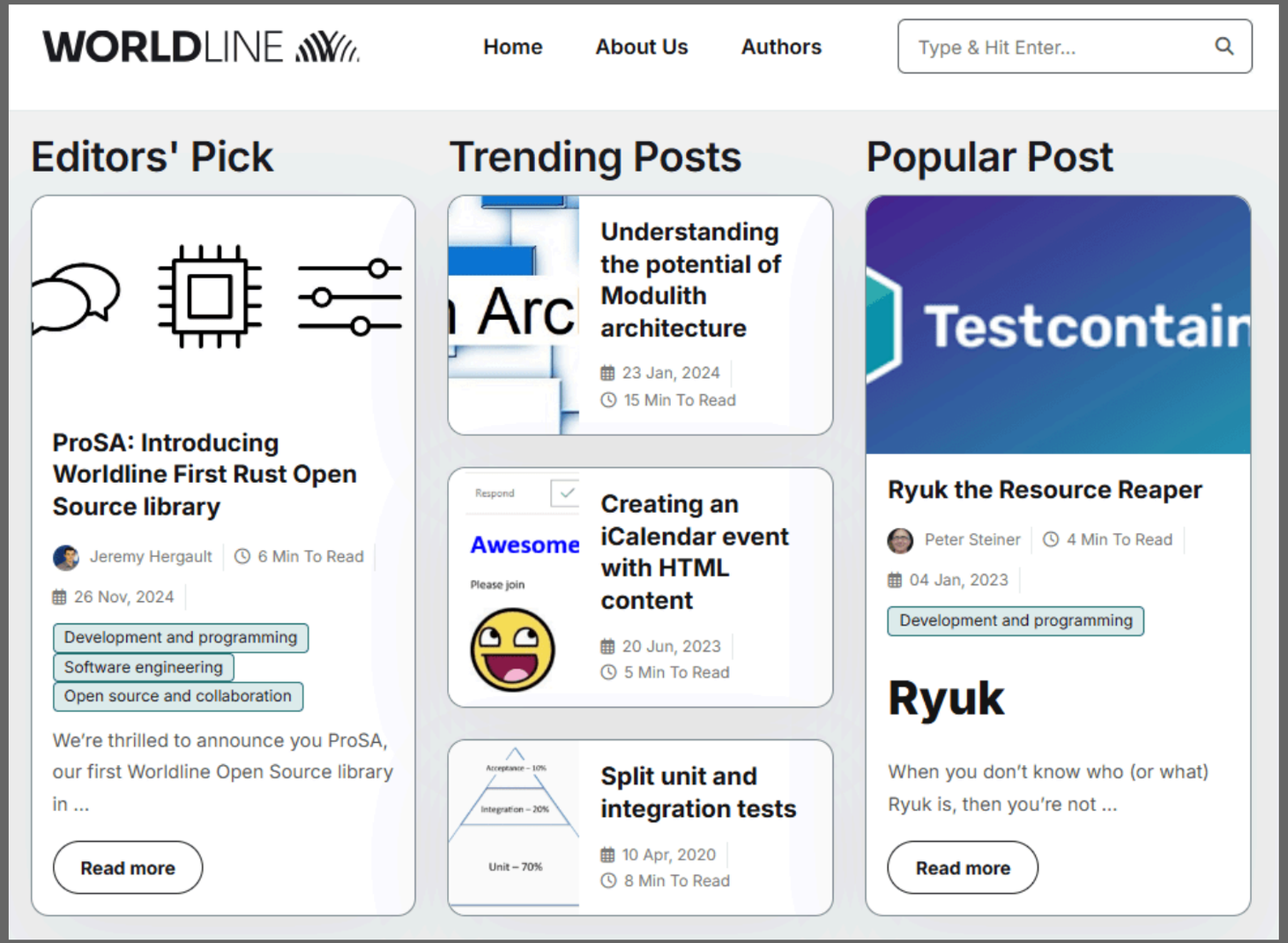
@yostane

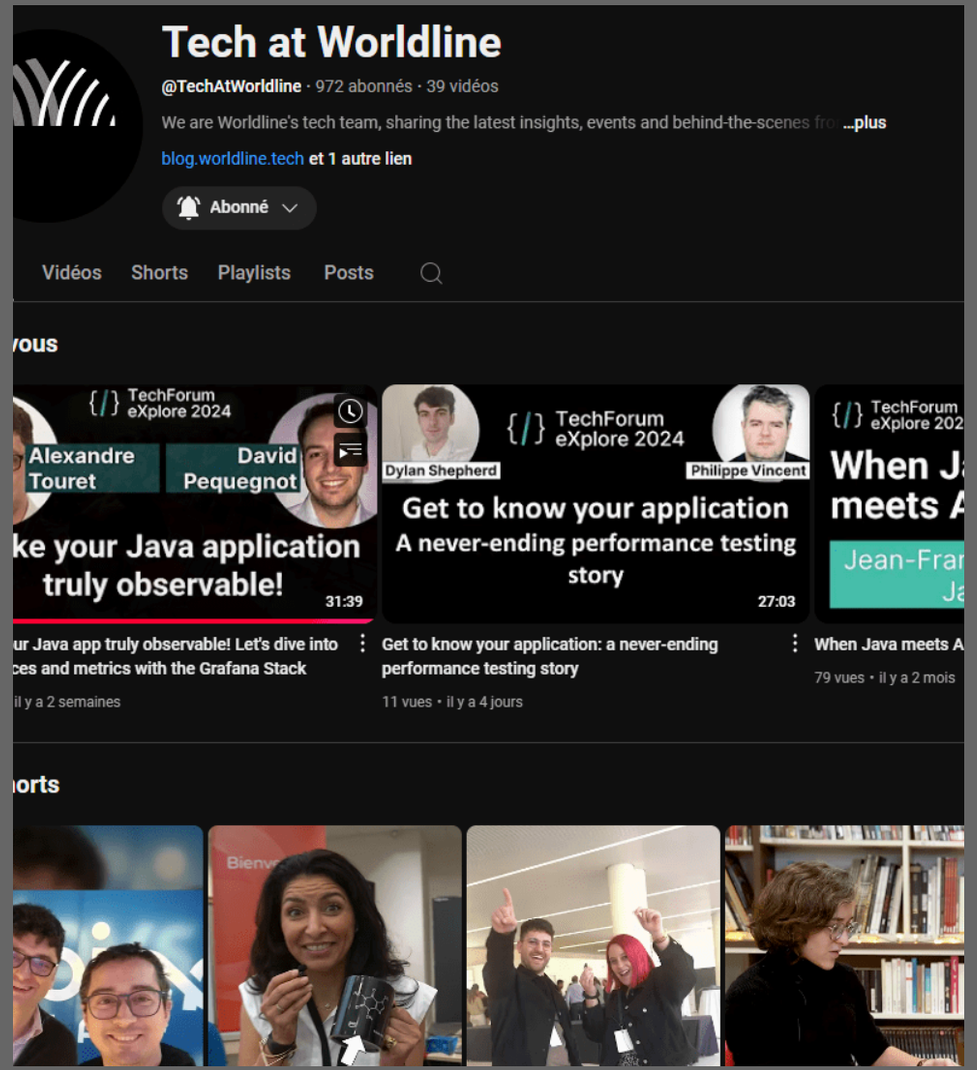
DevRel engineer @ Worldline

Teacher (yostane.github.io/lectures)

Lille Android User Group

Video game collector





blog.worldline.tech

@TechAtWorldline

@worldlinetech

@techatworldline.bsky.social

Prolog

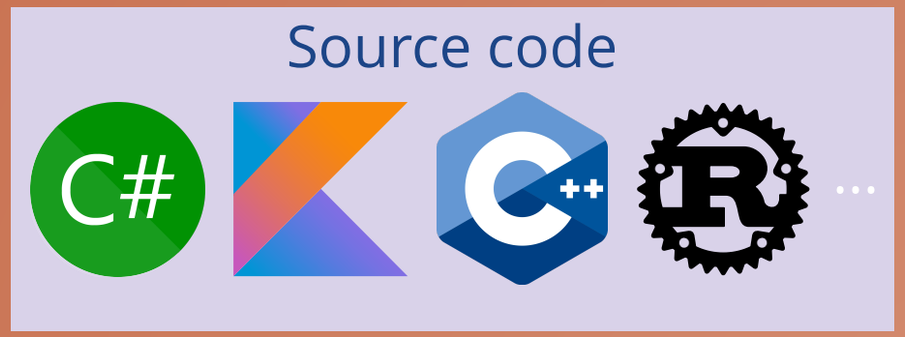
WebAssembly

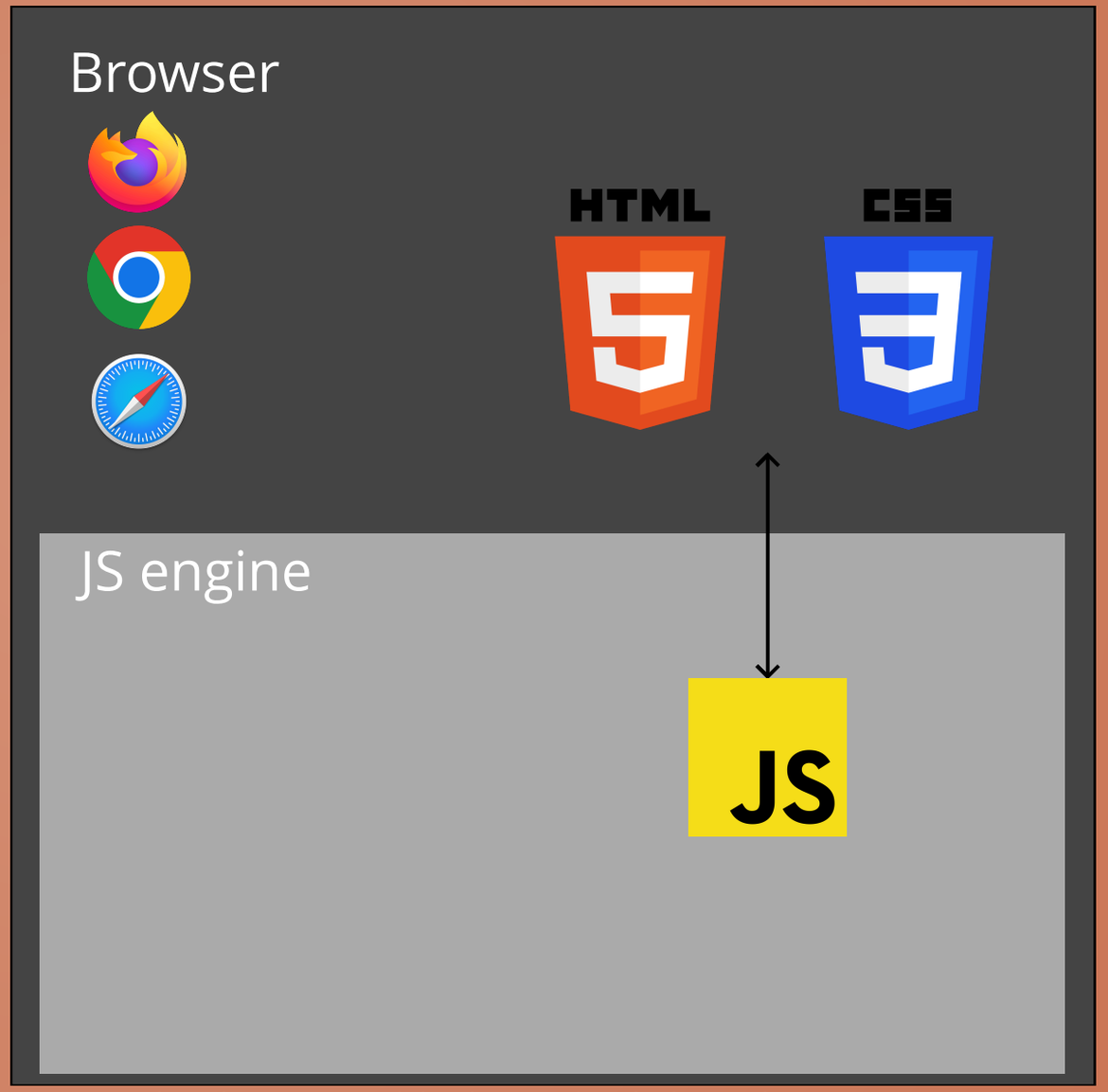
Web Assembly (WASM)

- Portable binary instruction format
- Initially targeted for web browsers (to speed up JS)


```
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.....|
```

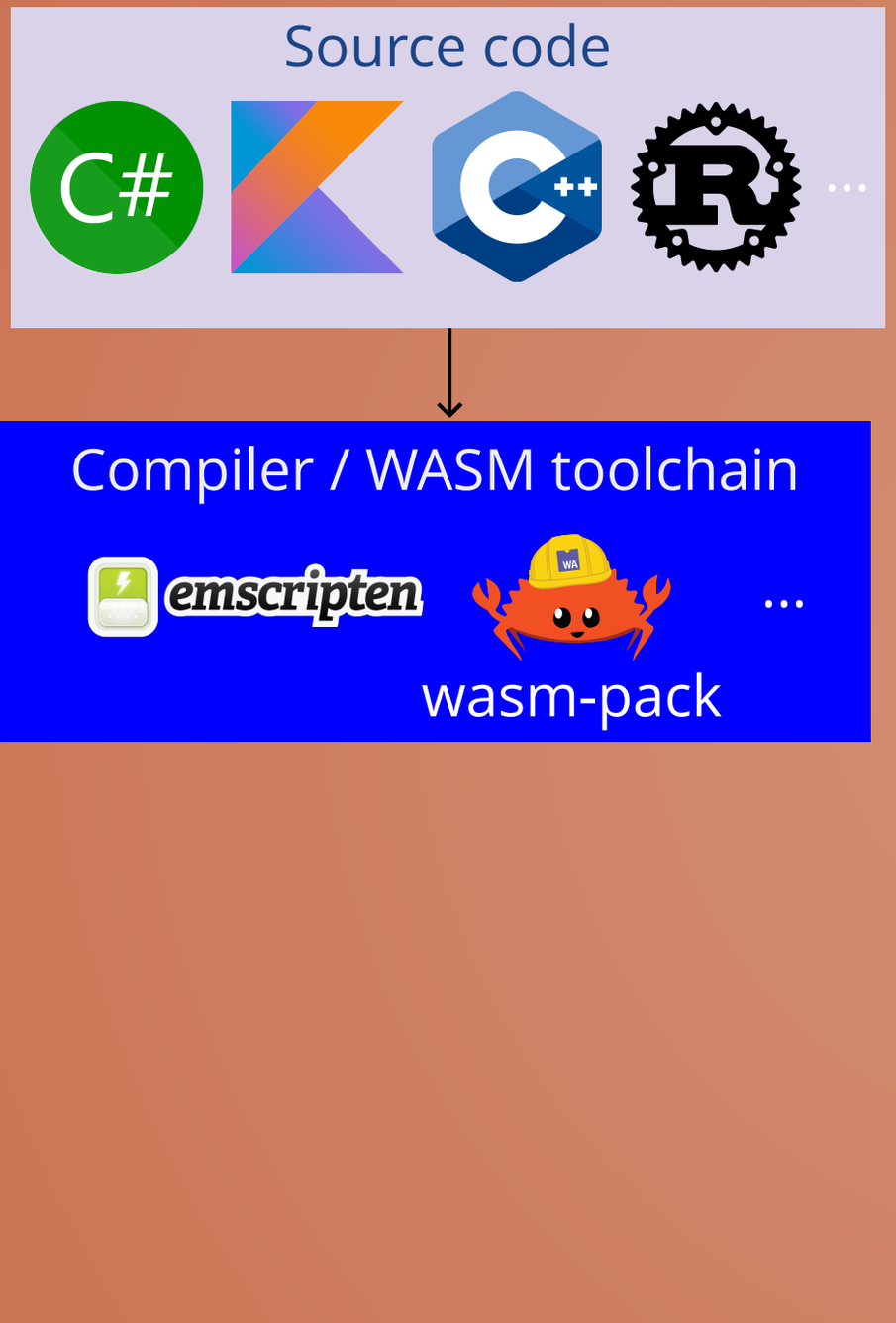
WASM on the web





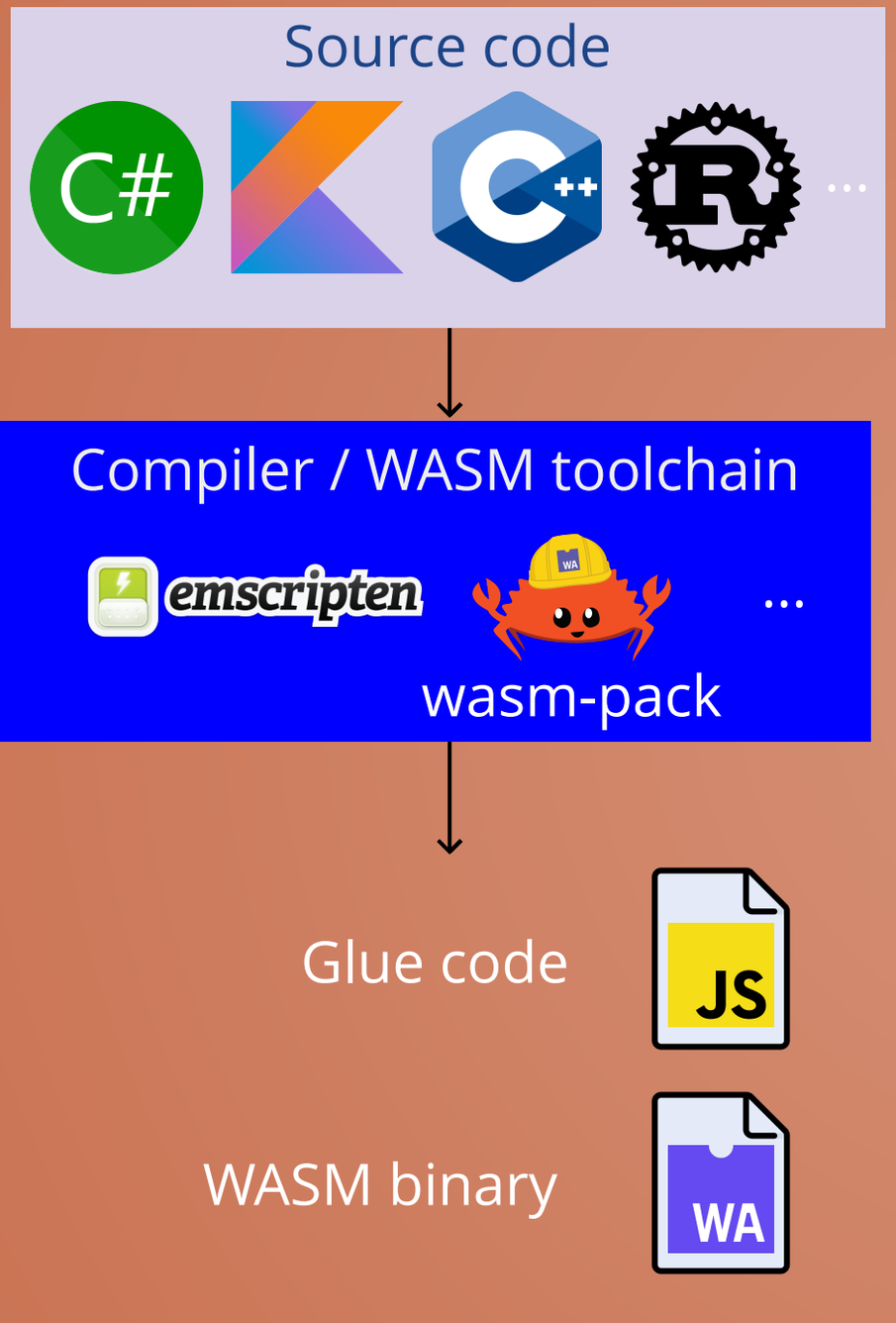
Based on: https://wasmlabs.dev/articles/docker-without-containers/

WASM on the web



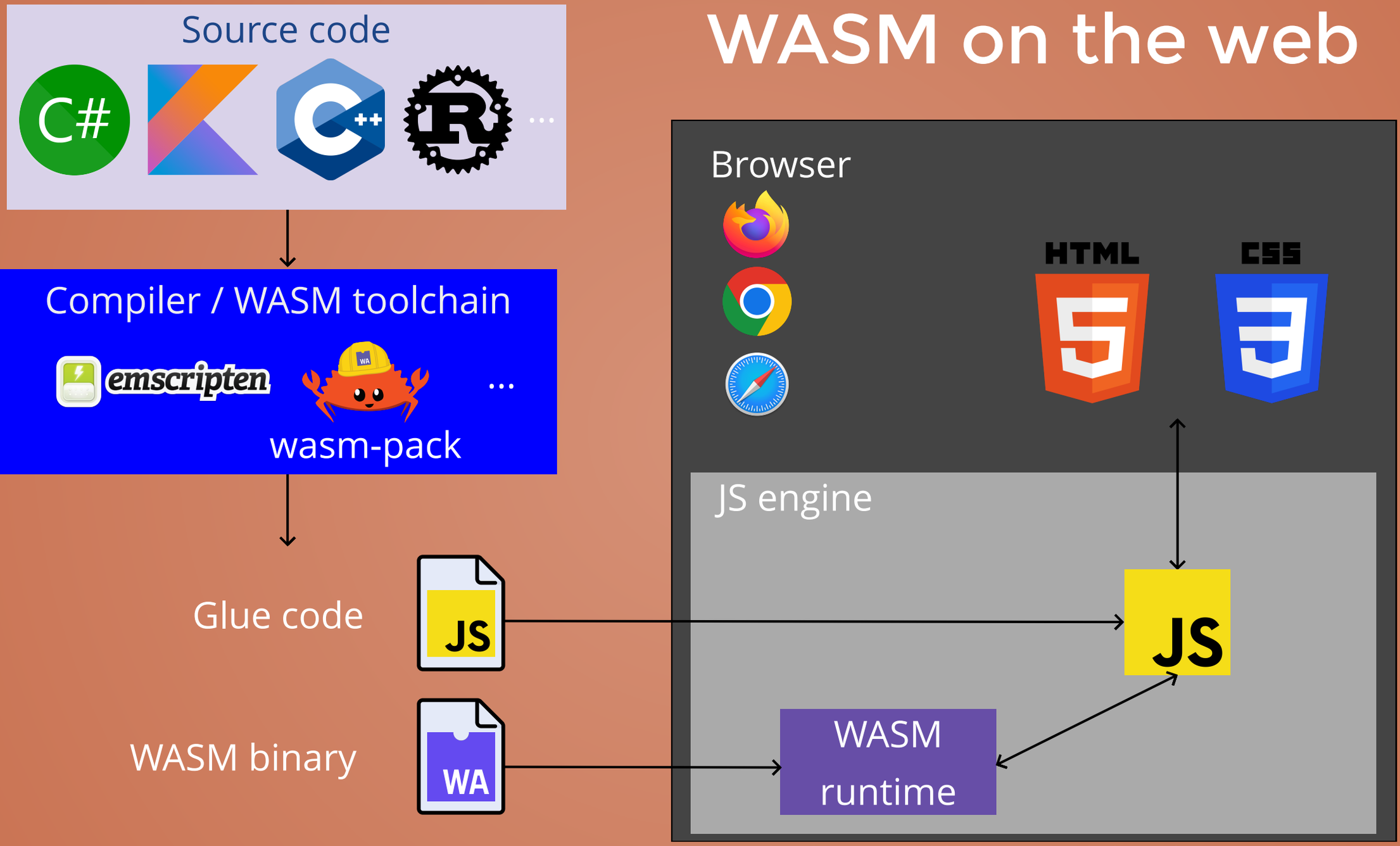
Based on: https://wasmlabs.dev/articles/docker-without-containers/

WASM on the web

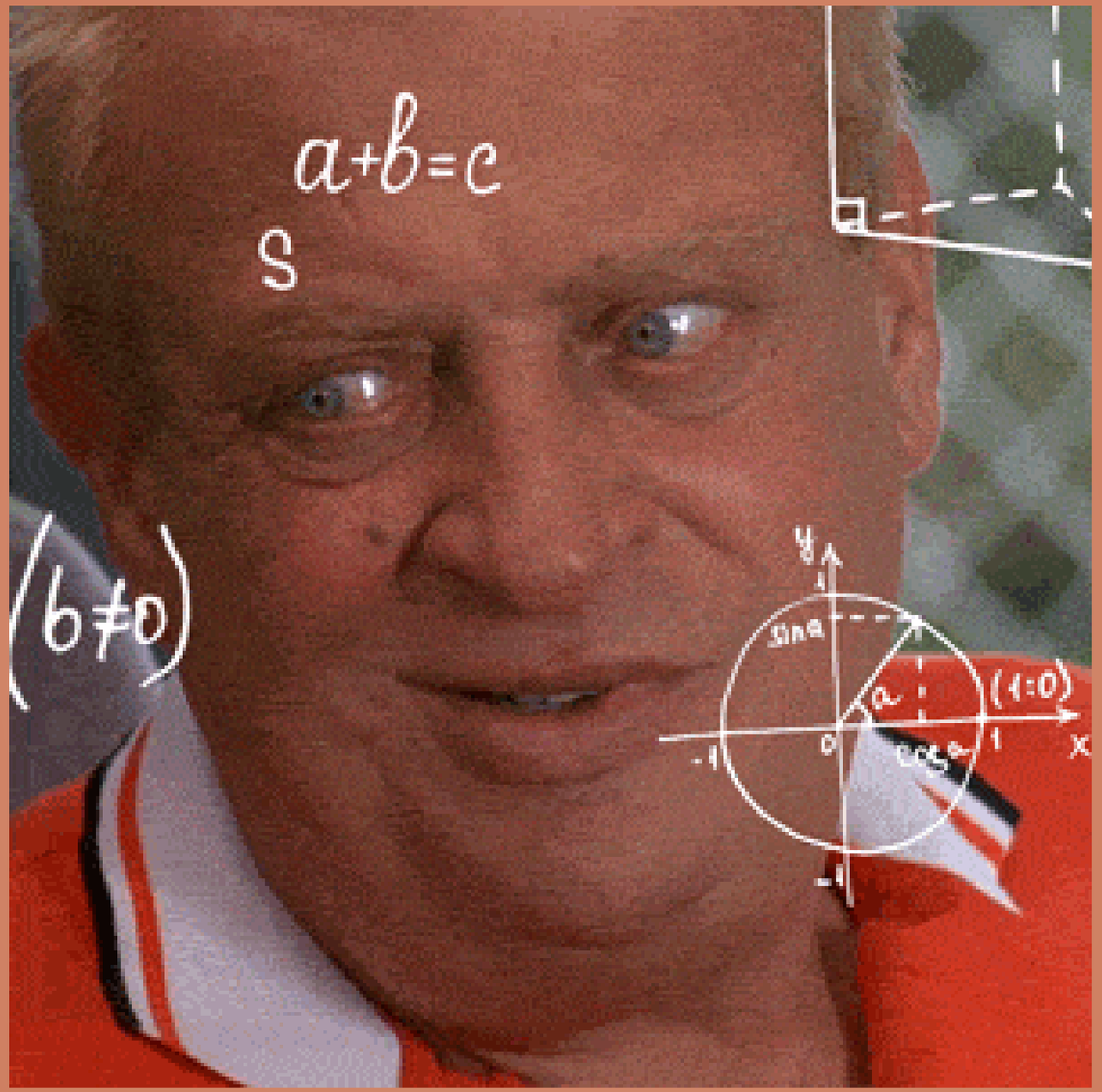


Based on: https://wasmlabs.dev/articles/docker-without-containers/

WASM on the web



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



Need to do something related to WASM

Chapter 1

.Net and WASM

.NET and WASM

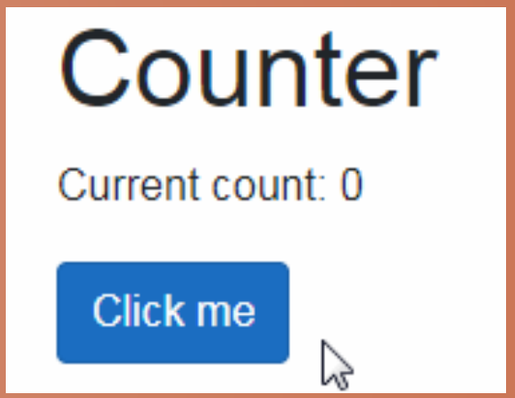
- .NET: C# OSS cross-platform framework

.NET and WASM

- .NET: C# OSS cross-platform framework
- Compiles to WASM
  - Blazor WASM: Client framework (like VueJS)
  - wasm-tools: Vanilla approach

Blazor wasm

```
1  @page "/counter"
2
3    <h1>Counter</h1>
4    <p>Current count: @currentCount</p>
5    <button @onclick="IncrementCount">Click me</button>
6
7    @code {
8        private int currentCount = 0;
9        private void IncrementCount() { currentCount++;
10   }
```



.NET 7+
wasm-tools

```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();
    }
```

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

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

.NET 7+
wasm-tools

```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();
    }
```

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

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

.NET 7+
wasm-tools

```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();
    }
```

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

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

dotnet wasmbrowser template project

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

MVG's video is a great source of inspiration


DOOM

1993 for DOS


100%
AMMO
HEALTH
ARMS
0%
ARMOR
BULL 23 / 200
SHEL 0 / 50
ROKT 0 / 50
CELL 0 / 300

1993 for DOS

Engine

1993 for DOS

+

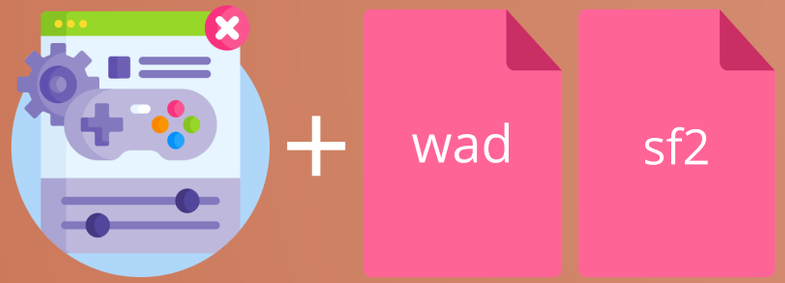
wad

sf2

Engine

Resources

1993 for DOS



Engine Resources

🌟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/

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**

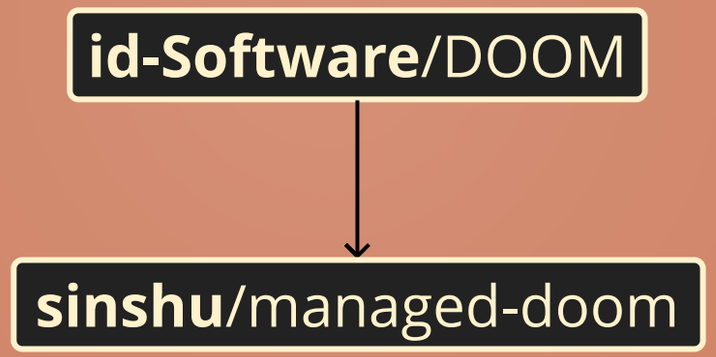
ManagedDoom

- **.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

Porting plan

- Clone **sininshu/managed-doom**
- Change the build target to **WASM**
- Re-implement SFML code:

dall-e

Porting plan

- Clone **sininshu/managed-doom**
- Change the build target to **WASM**
- Re-implement SFML code:
    1. With mocks until the project compiles

dall-e

Porting plan

- Clone **sininshu/managed-doom**
- Change the build target to **WASM**
- Re-implement SFML code:
  1. With mocks until the project compiles
  2. Then replace mocks with correct implementations

dall-e

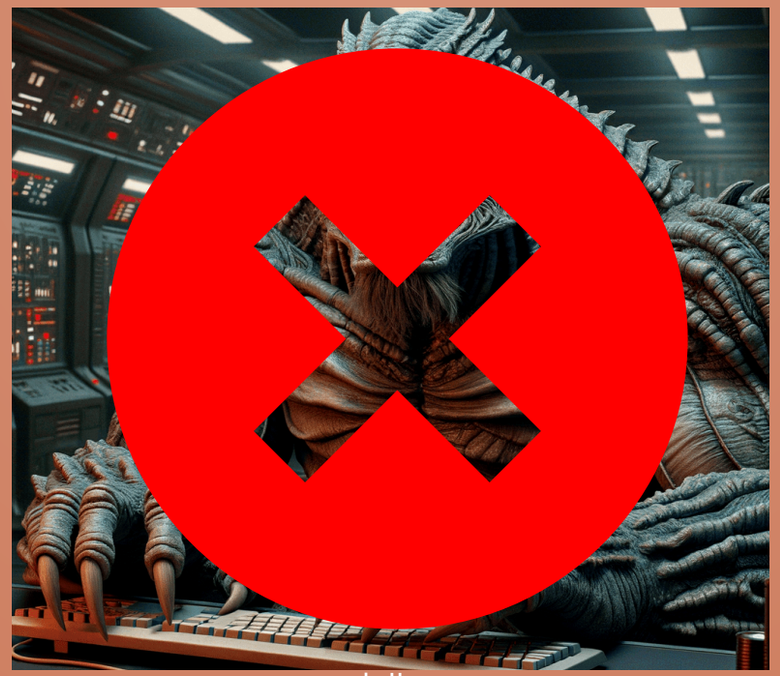
Porting plan

- Clone **sininshu/managed-doom**
- Change the build target to **WASM**
- Re-implement SFML code:
  1. With mocks until the project compiles
  2. Then replace mocks with correct implementations
- Implemenet minimal JS (gameloop, audio and video rendering)

dall-e

Porting plan

- Clone **sininshu/managed-doom**
- Change the build target to **WASM**
- Re-implement SFML code:
  1. With mocks until the project compiles
  2. Then replace mocks with correct implementations
- Implemenet minimal JS (gameloop, audio and video rendering)



dall-e

Porting plan

- Clone **sininshu/managed-doom**
- Change the build target to **WASM**
- Re-implement SFML code:
  1. With mocks until the project compiles
  2. Then replace mocks with correct implementations
- Implemenet minimal JS (gameloop, audio and video rendering)

dall-e

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);
```

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 implement it first with a mock

```
 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    }
```

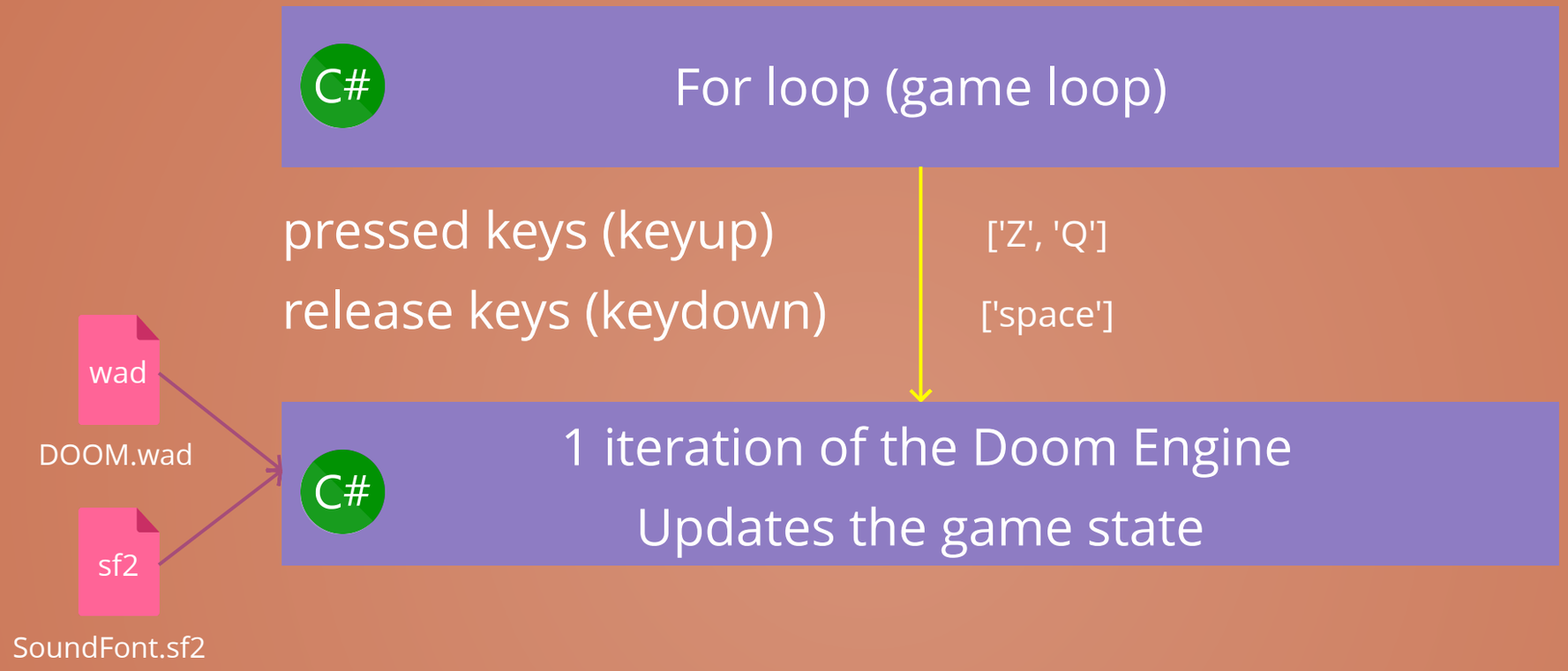
After that, with a complete implementation

```
 1 namespace SFML.Audio
 2   {
 3       public class SoundBuffer
 4       {
 5           public short[] samples;
 6           private int v;
 7           public uint sampleRate;
 8
 9           public SoundBuffer(short[] samples, int v, uint sampleRate)
10           {
11               this.samples = samples;
12               this.v = v;
13               this.sampleRate = sampleRate;
14           }
15
16           public Time Duration { get; internal set; }
17       }
18   }
```

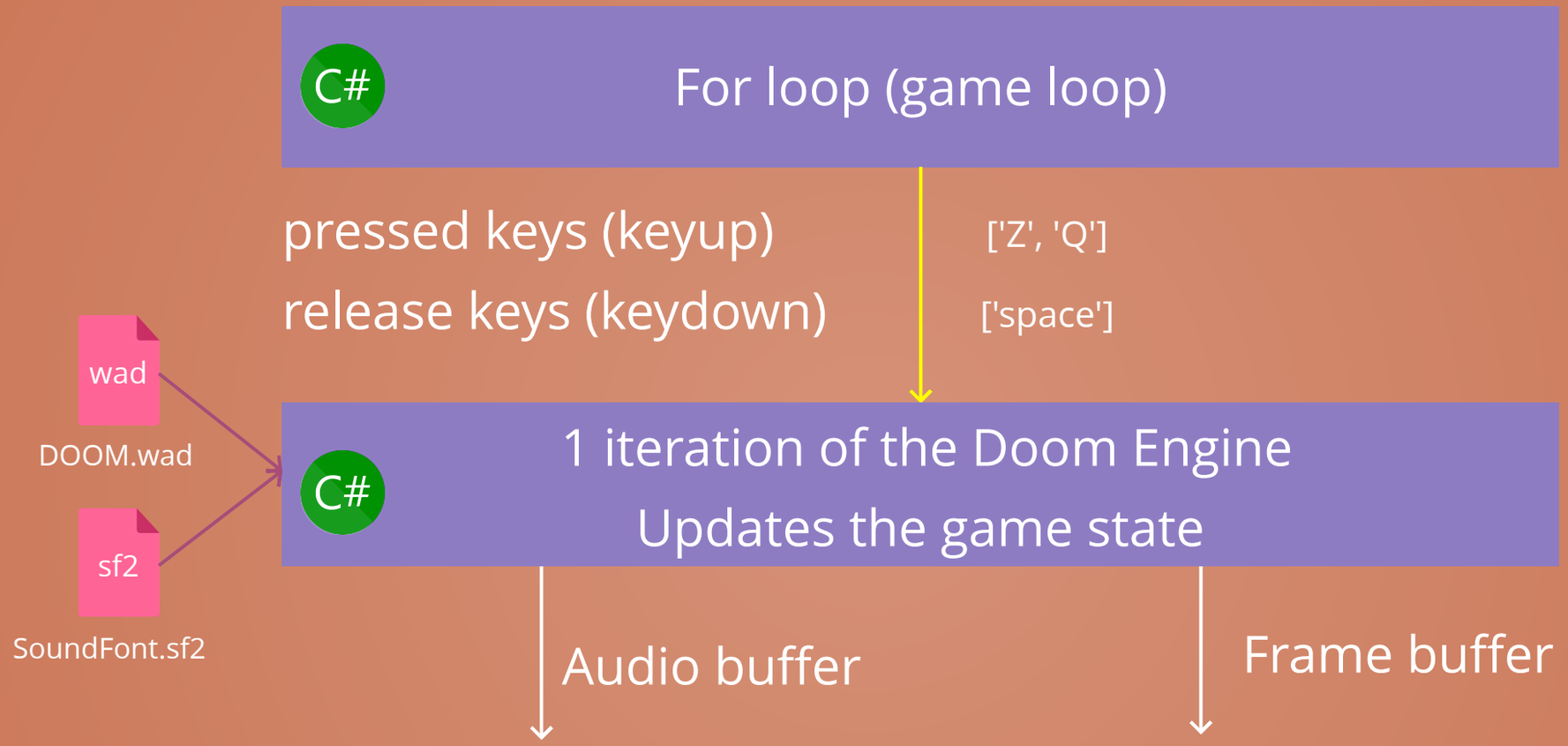
ManagedDoom V1 architecture

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

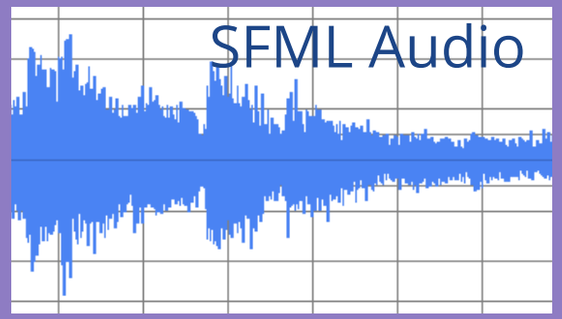
C# 1 iteration of the Doom Engine

Updates the game state

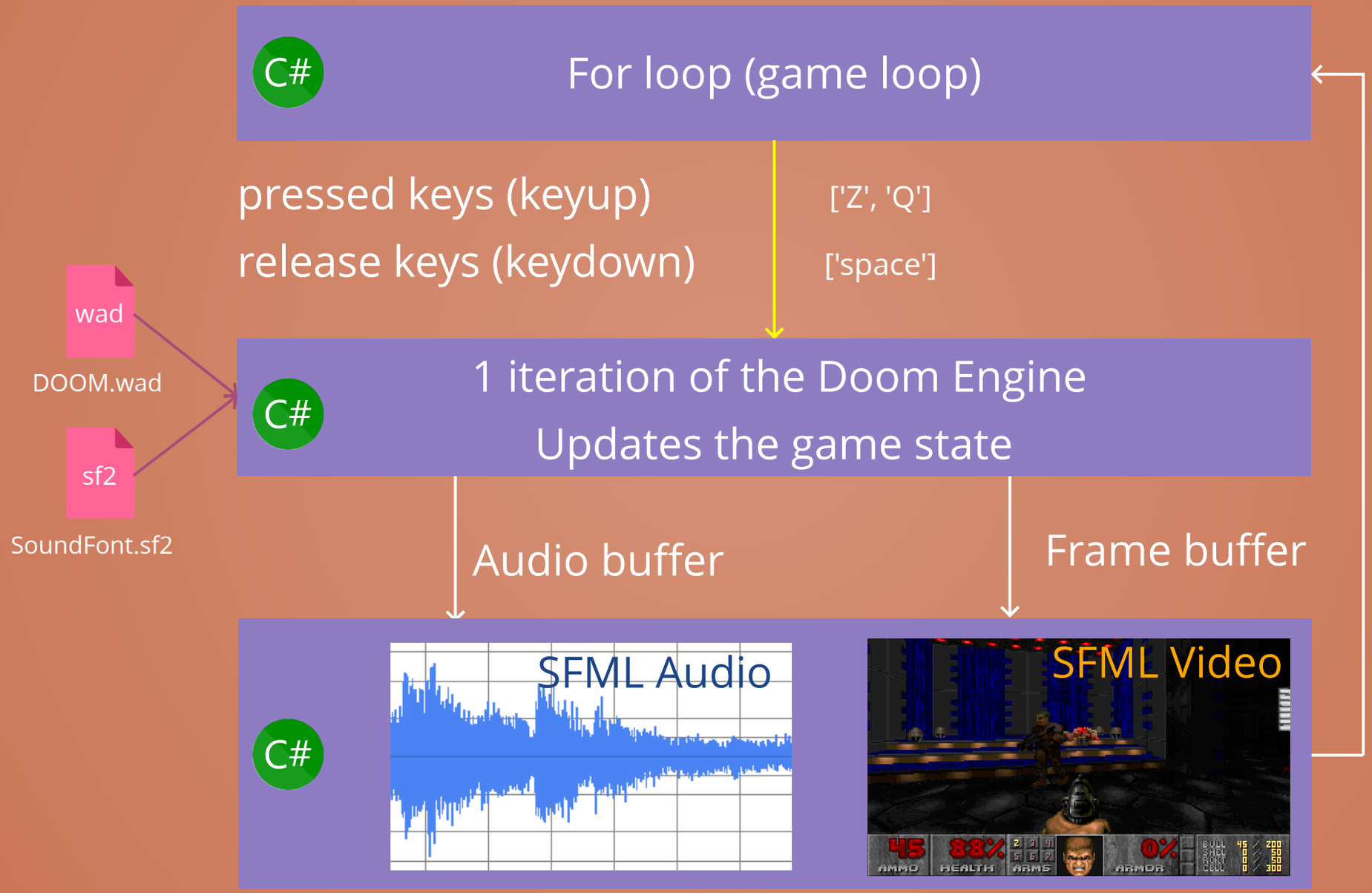
Audio buffer

Frame buffer

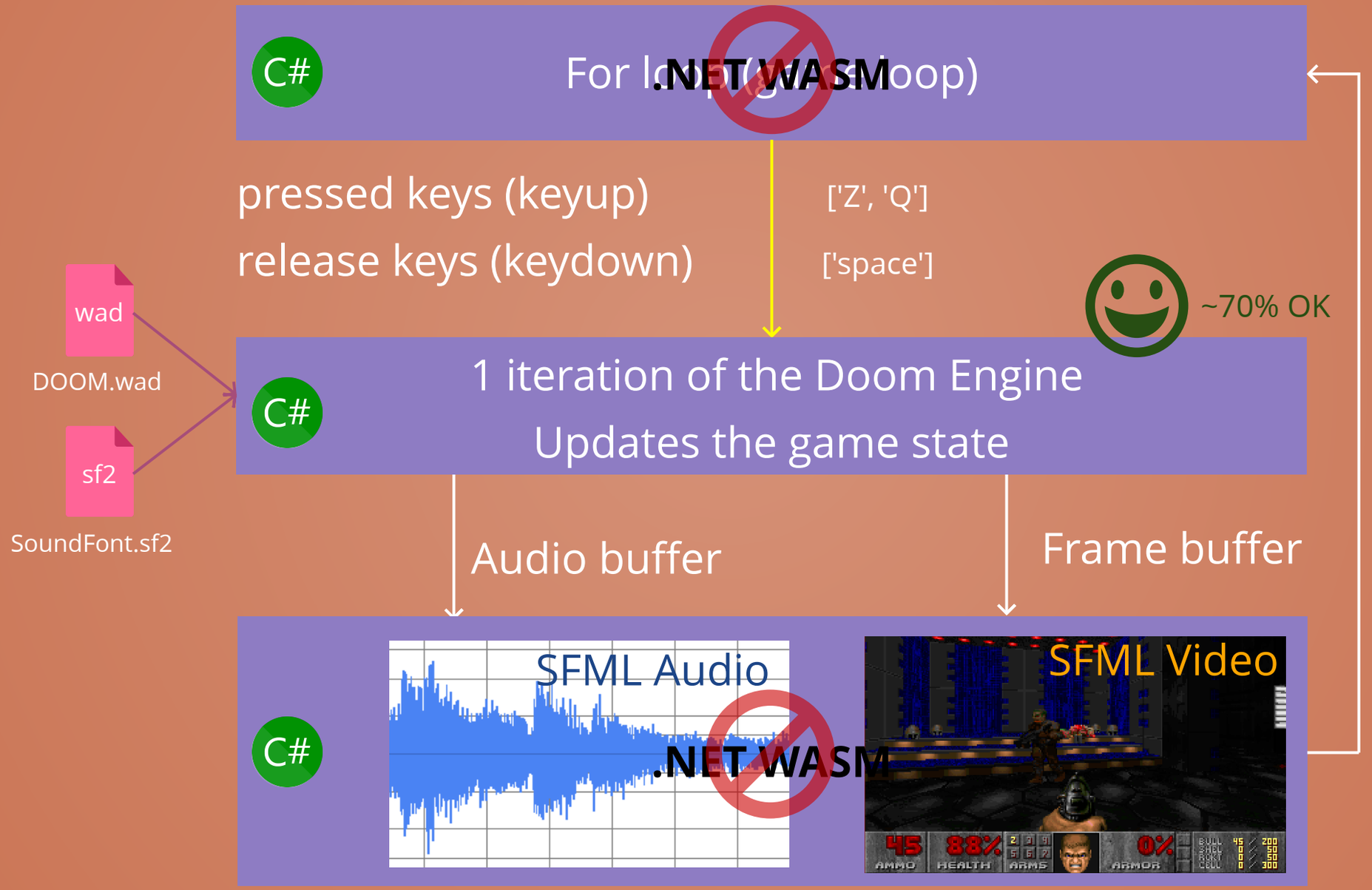
C#



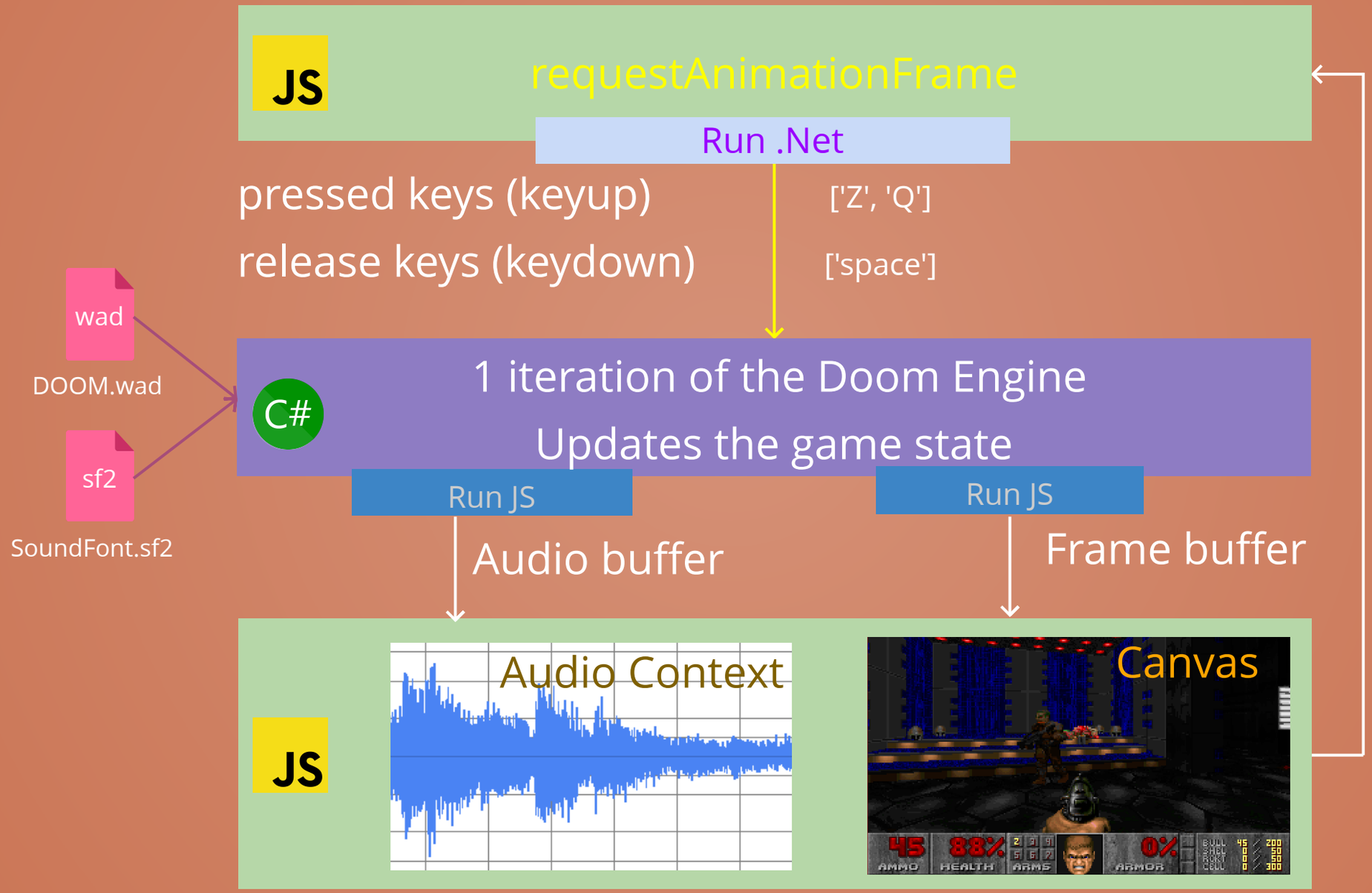
ManagedDoom V1 architecture



ManagedDoom V1 architecture



Blazor Doom architecture



Chapter 4

Knee-deep in the code

Chapter 4

Knee-deep in the code

Entry point

```
<html>
    <head>
      <!-- Sets .Net interop and starts the game loop -->
      <script type="module" src="./main.js"></script>
     </head>
    <body>
      <canvas id="canvas" width="320" height="200"
              style="image-rendering: pixelated" />
    </body>
  </html>
```

Entry point

```html
<html>
    <head>
      <!-- Sets .Net interop and starts the game loop -->
      <script type="module" src="./main.js"></script>
     </head>
    <body>
      <canvas id="canvas" width="320" height="200"
              style="image-rendering: pixelated" />
    </body>
  </html>
```

Entry point

```
<html>
    <head>
      <!-- Sets .Net interop and starts the game loop -->
      <script type="module" src="./main.js"></script>
     </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);
  }
```

JS

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);
  }
```

JS

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);
  }
```

JS

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);
  }
```

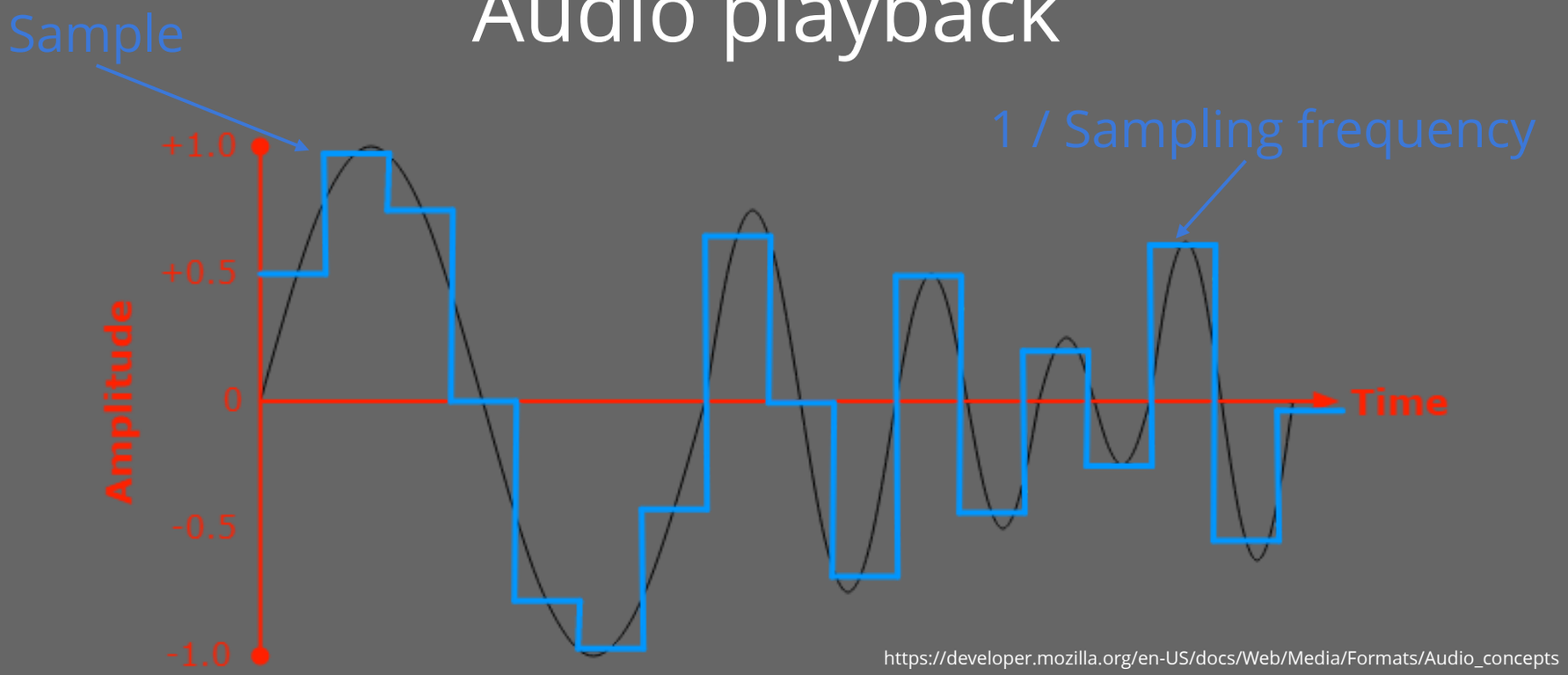
JS

```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);
    }
  }
```

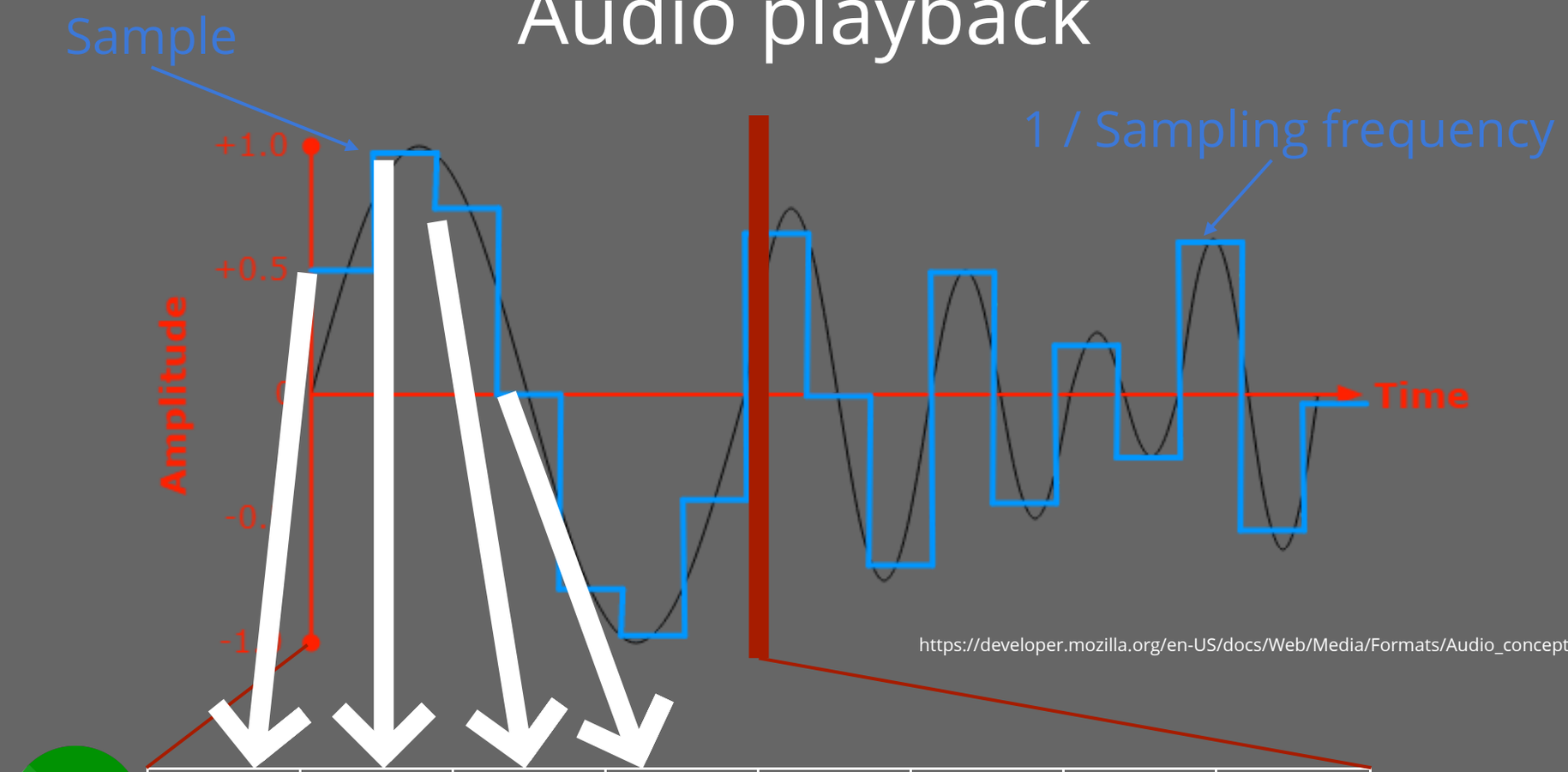
C#

Audio playback



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

Audio playback

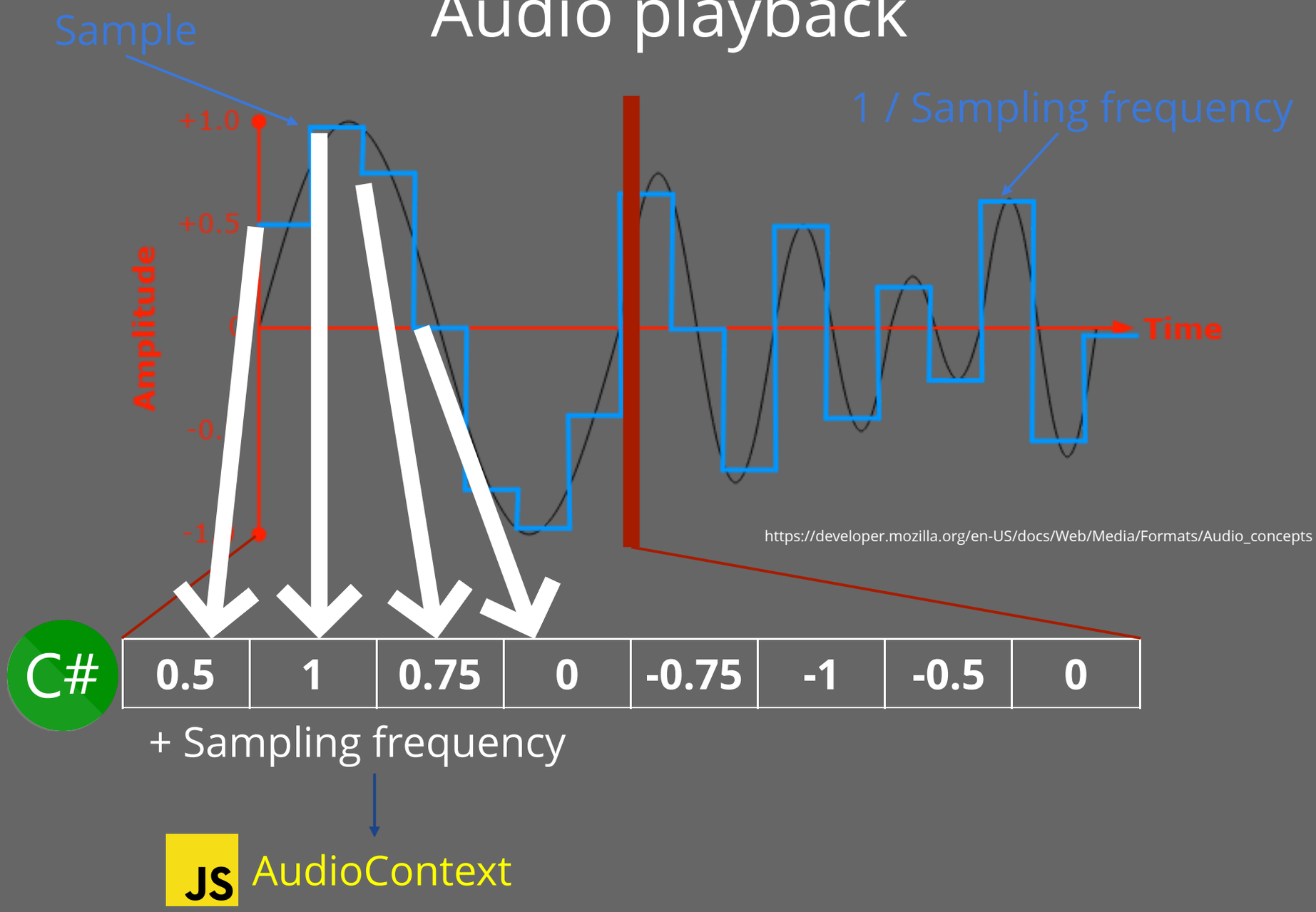


C#

0.5	1	0.75	0	-0.75	-1	-0.5	0

+ Sampling frequency

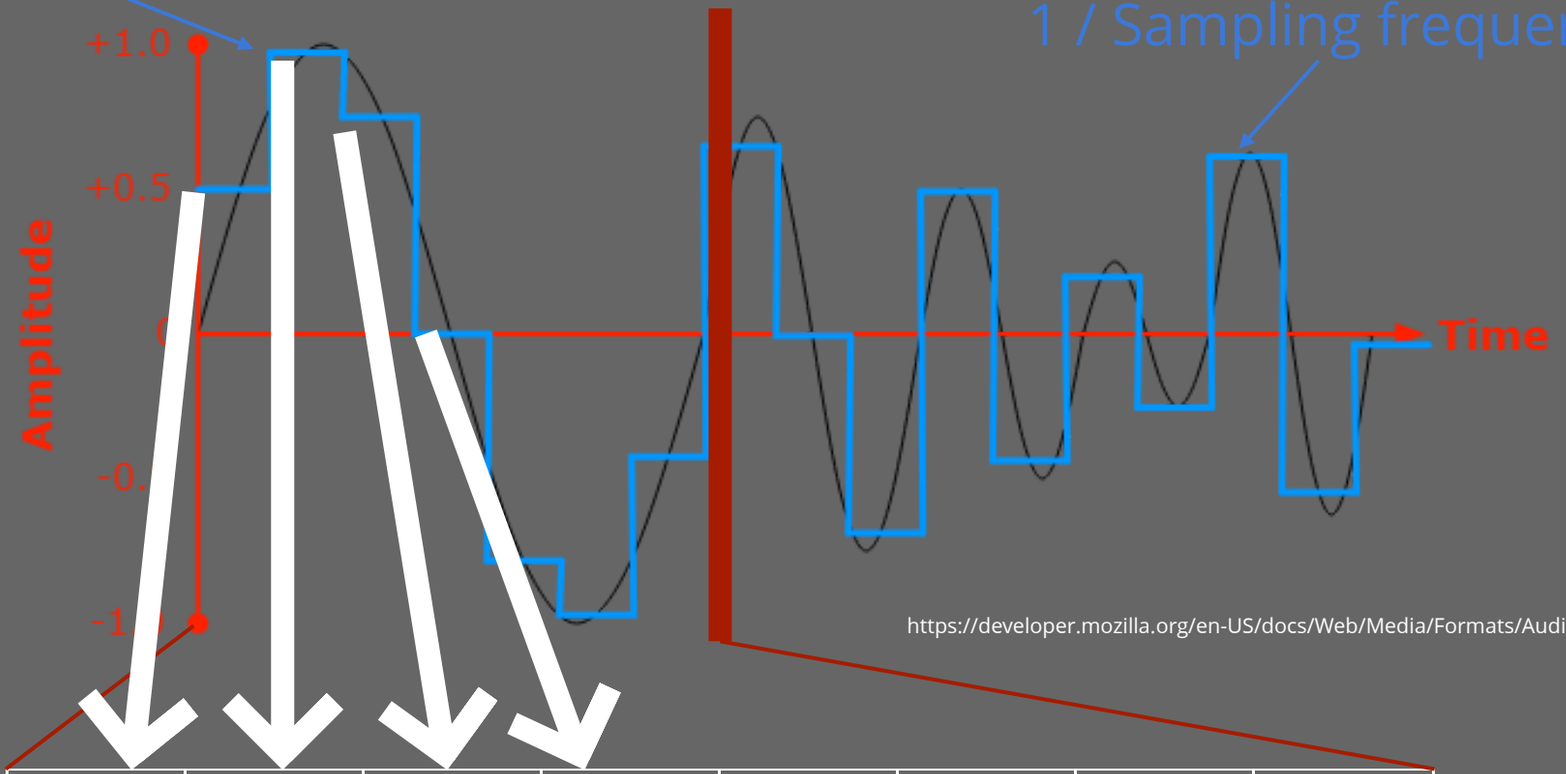
Audio playback



Audio playback

Sample

1 / Sampling frequency



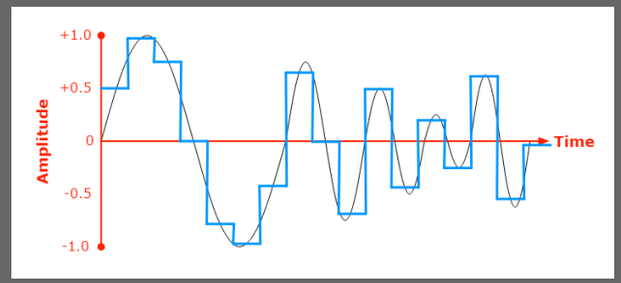
https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

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

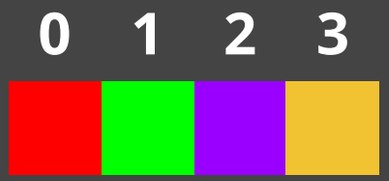
From 1D frame to a 2D frame

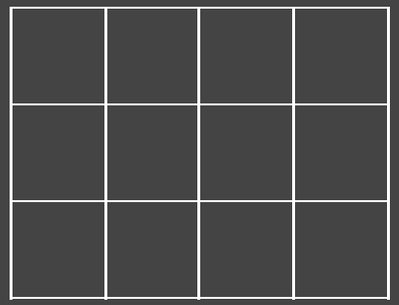
Frame data

12 bytes

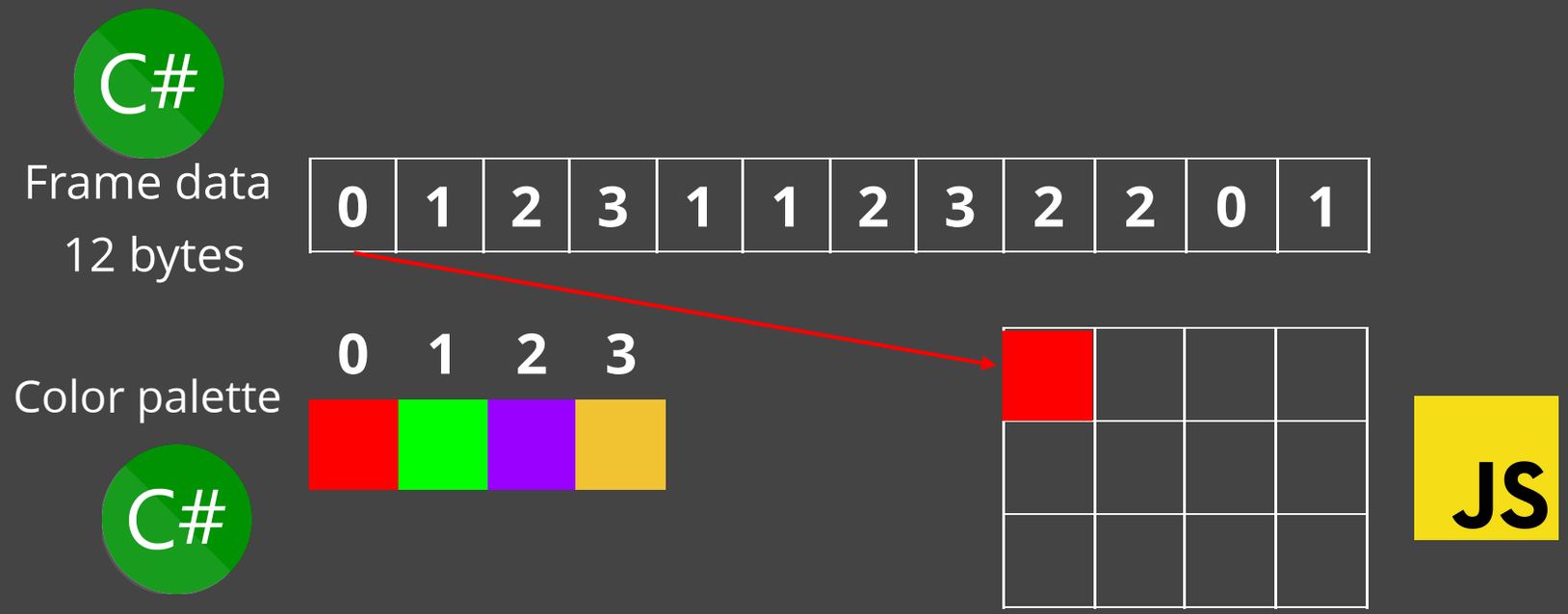
0	1	2	3	1	1	2	3	2	2	0	1

Color palette

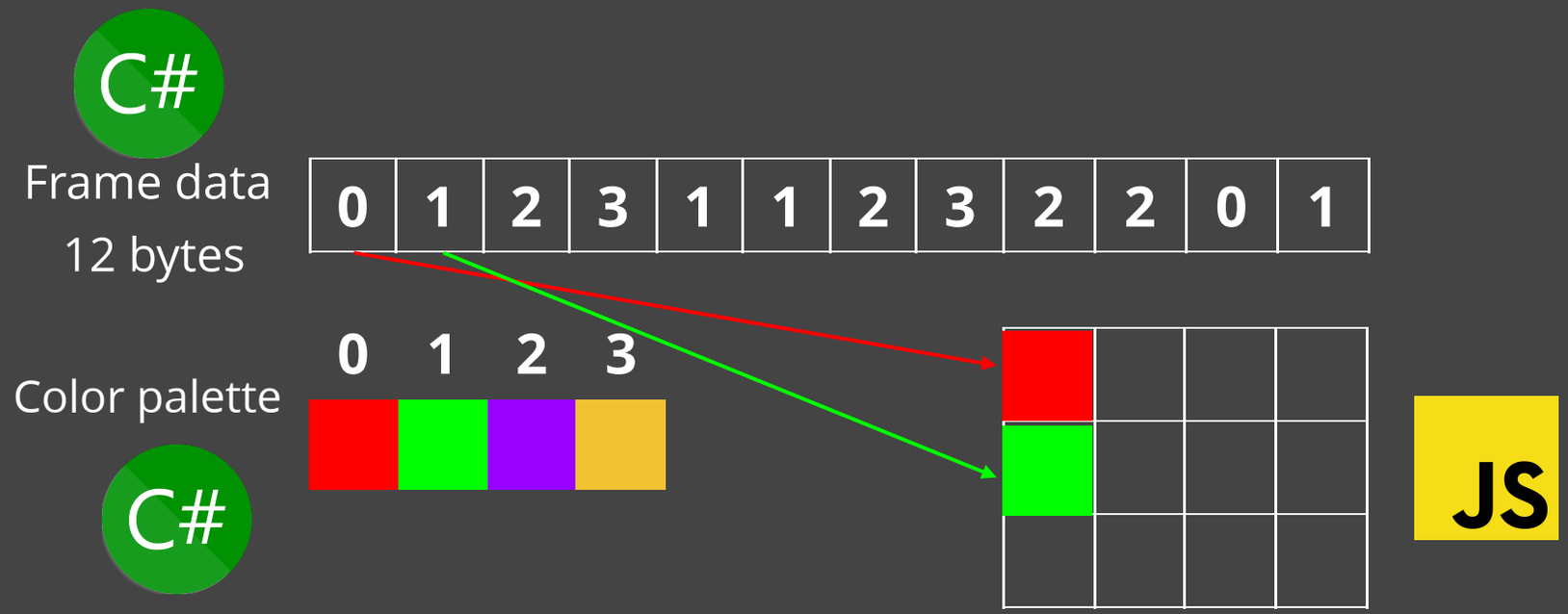




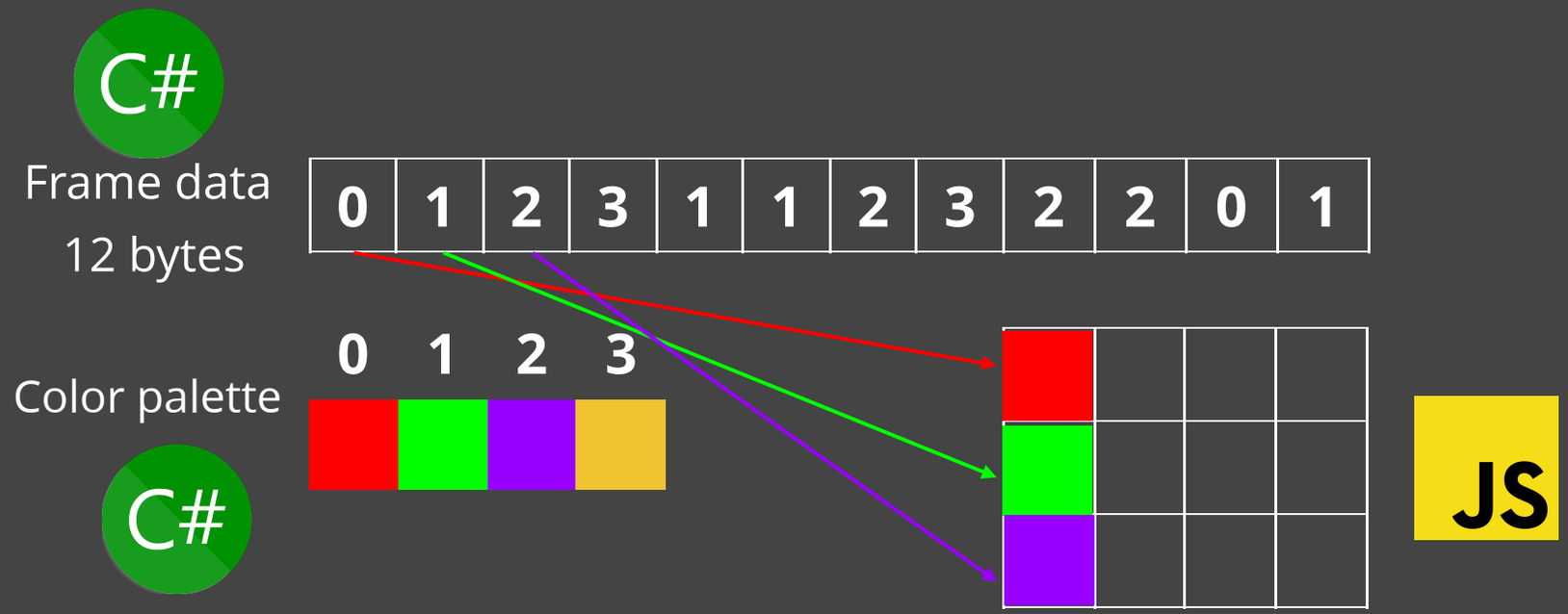
From 1D frame to a 2D frame



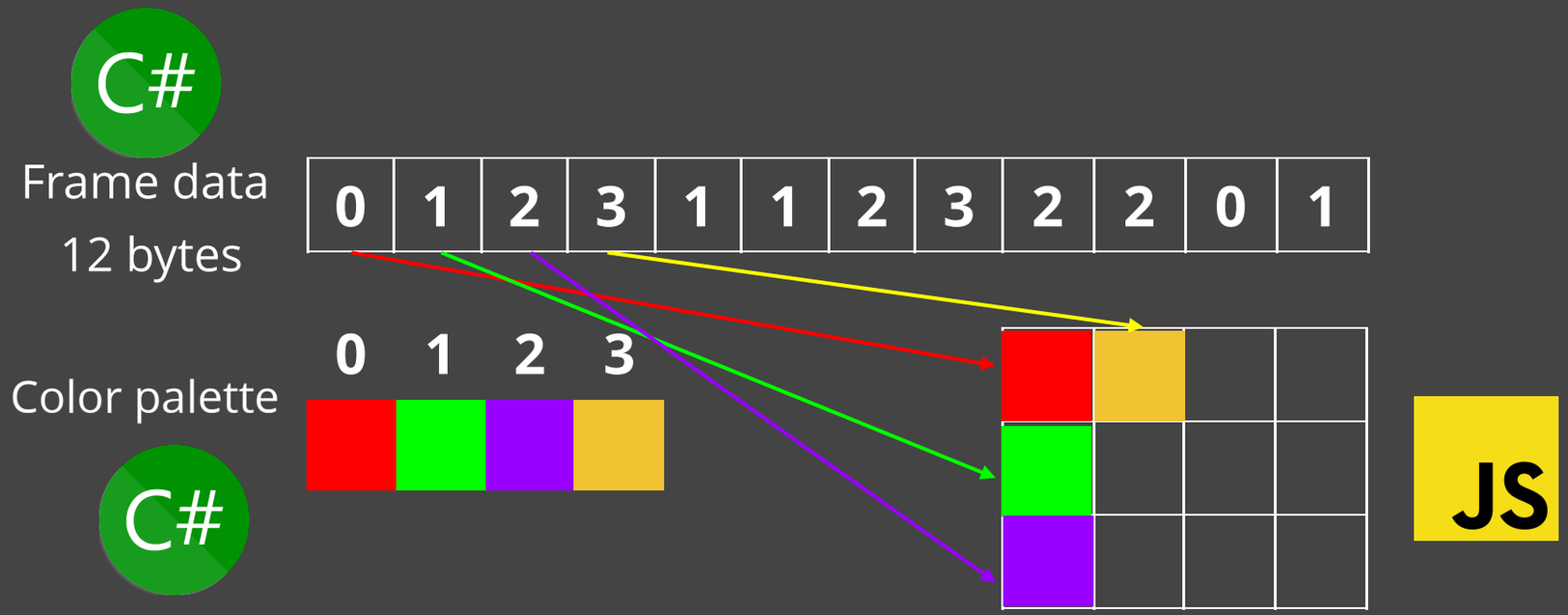
From 1D frame to a 2D frame



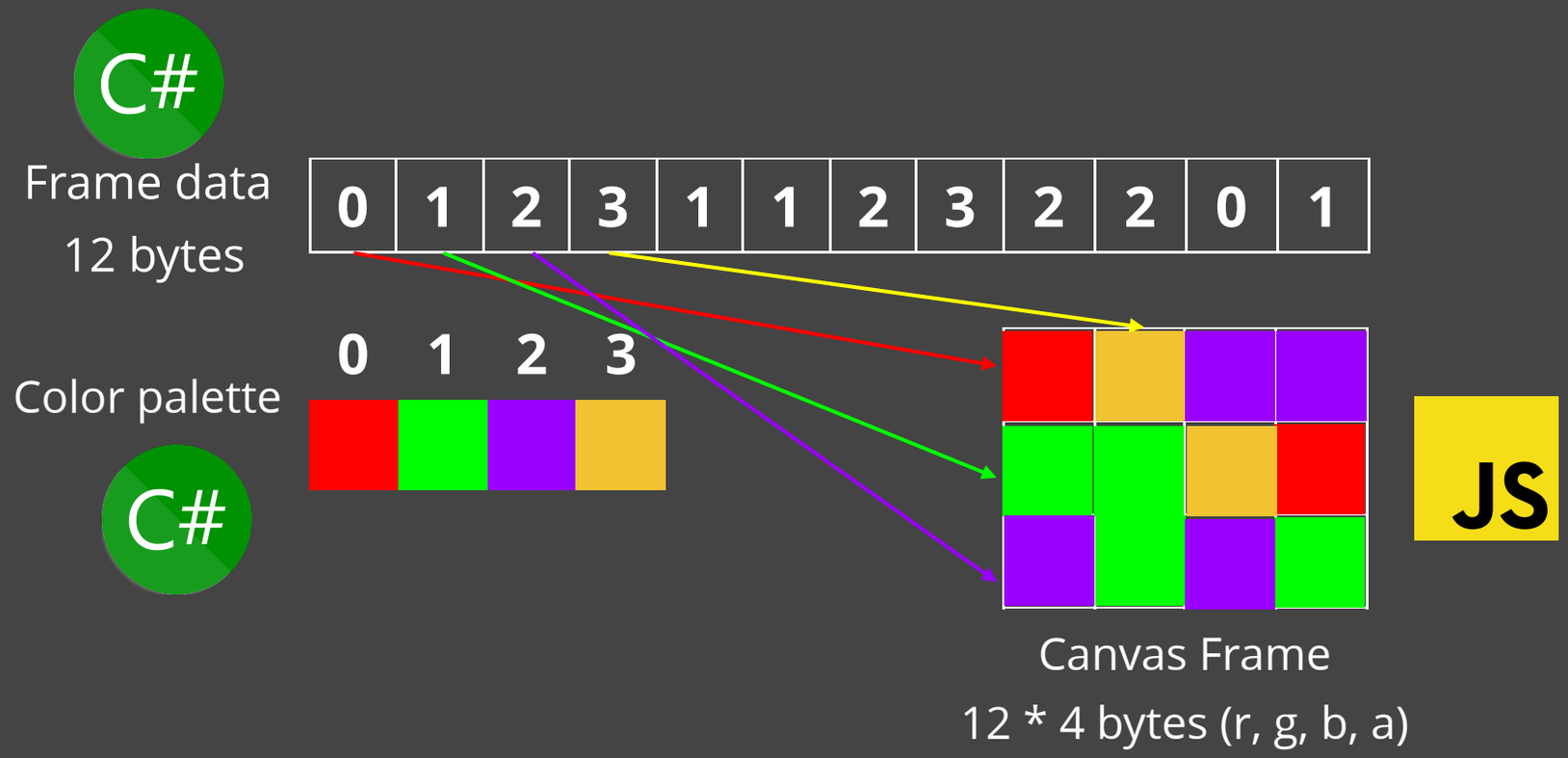
From 1D frame to a 2D frame



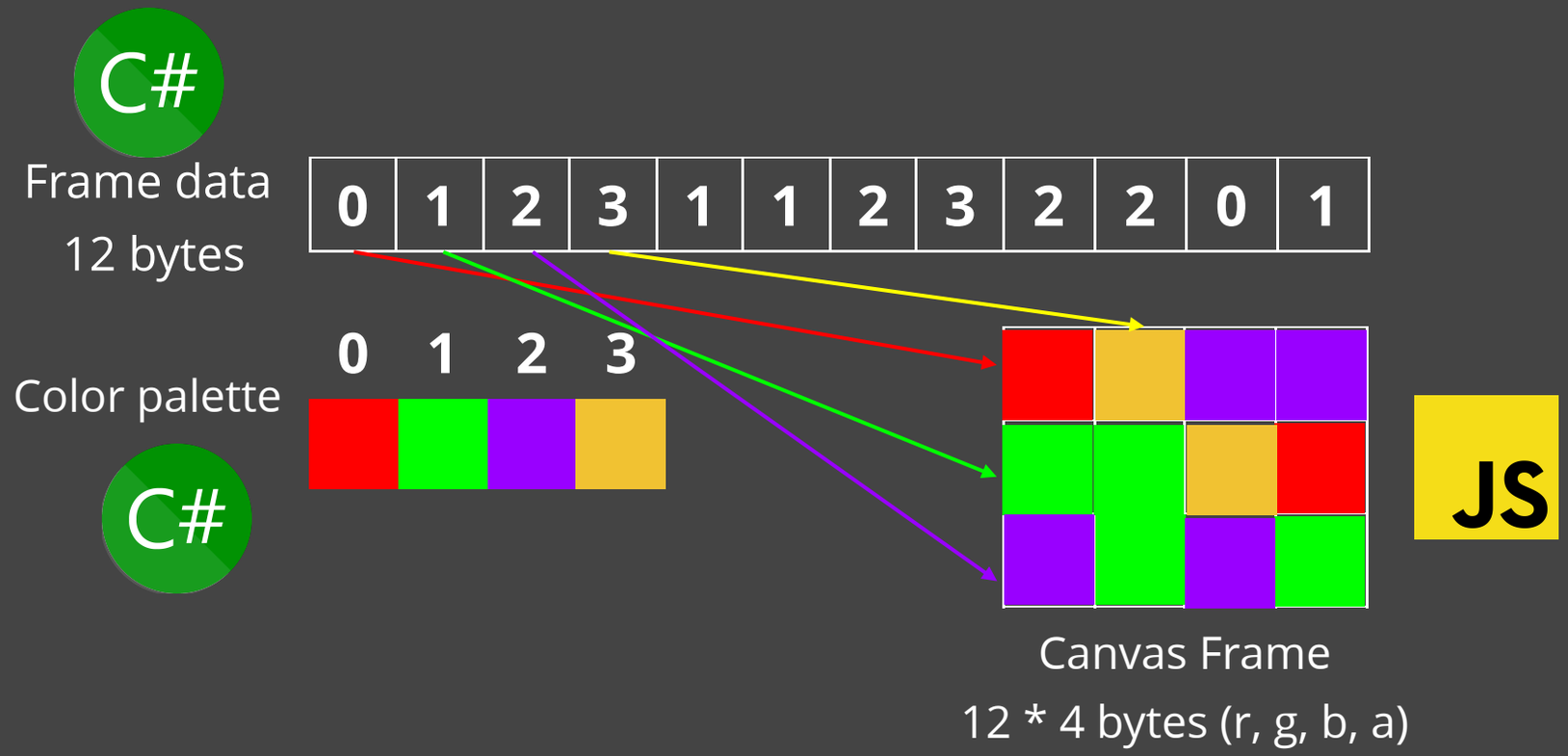
From 1D frame to a 2D frame



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

```csharp
public class DoomRenderer
  {
    // Called by updateGameState
    private void Display(uint[] colors)
    {
      BlazorDoom.Renderer.renderOnJS(screen.Data, (int[])((object)colors));
    }
  }
```


```csharp
namespace BlazorDoom
  {
      [SupportedOSPlatform("browser")]
      public partial class Renderer
      {
          [JSImport("drawOnCanvas", "blazorDoom/renderer.js")]
          internal static partial string renderOnJS(byte[] screenData,
                                                    int[] colors);
      }
  }
```


```javascript
export function drawOnCanvas(screenData, colors) {
    //
  }
```

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) {
    // Get color from the color palette
    const color = colors[colorIndex];
    // Extract RGB and spread it in the canvas
    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
}
```

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) {
    // Get color from the color palette
    const color = colors[colorIndex];
    // Extract RGB and spread it in the canvas
    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
}
```

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) {
    // Get color from the color palette
    const color = colors[colorIndex];
    // Extract RGB and spread it in the canvas
    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
}
```

CHAPTER 5

DEMO(N)


Open doors
Shoot / validate
30 FPS - 33ms - 19ms
Tap Sound
Doom 1 shareware
Load WAD
Toggle max FPS
F
O
E
50
100%
AMMO
HEALTH
ARMS
0%
ARMOR
BULL 50 / 200
SHEL 0 / 50
ROKT 0 / 50
CELL 0 / 300
Move
WAD selection

CLOSING

KEY TAKEAWAYS

Key takeaways

Key takeaways

- WASM possibilities are limitless

Key takeaways

- WASM possibilities are limitless
- Porting a game is way to learn programming while having **fun**

SLIDE DECK

SLIDES OVER

SOURCE CODE

THANKS FOR PLAYING !
ANY QUESTION ?

Feedback