

RUN YOUR FAVORITE
GAMES EVERYWHERE
WITH WASM

THE BLAZOR DOOM
USE CASE

DEVOXX[®]

GENESIS
2024



How it started



How it's going





Yassine
Benabbas



DevRel / Teacher
Video game collector
😍 Kotlin, WASM, or any fancy technology

WORLDLINE 

#TechAtWorldline



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

PROTOTYPES
WebAssembly

Web Assembly (WASM)

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



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	72getUniver
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....|
```



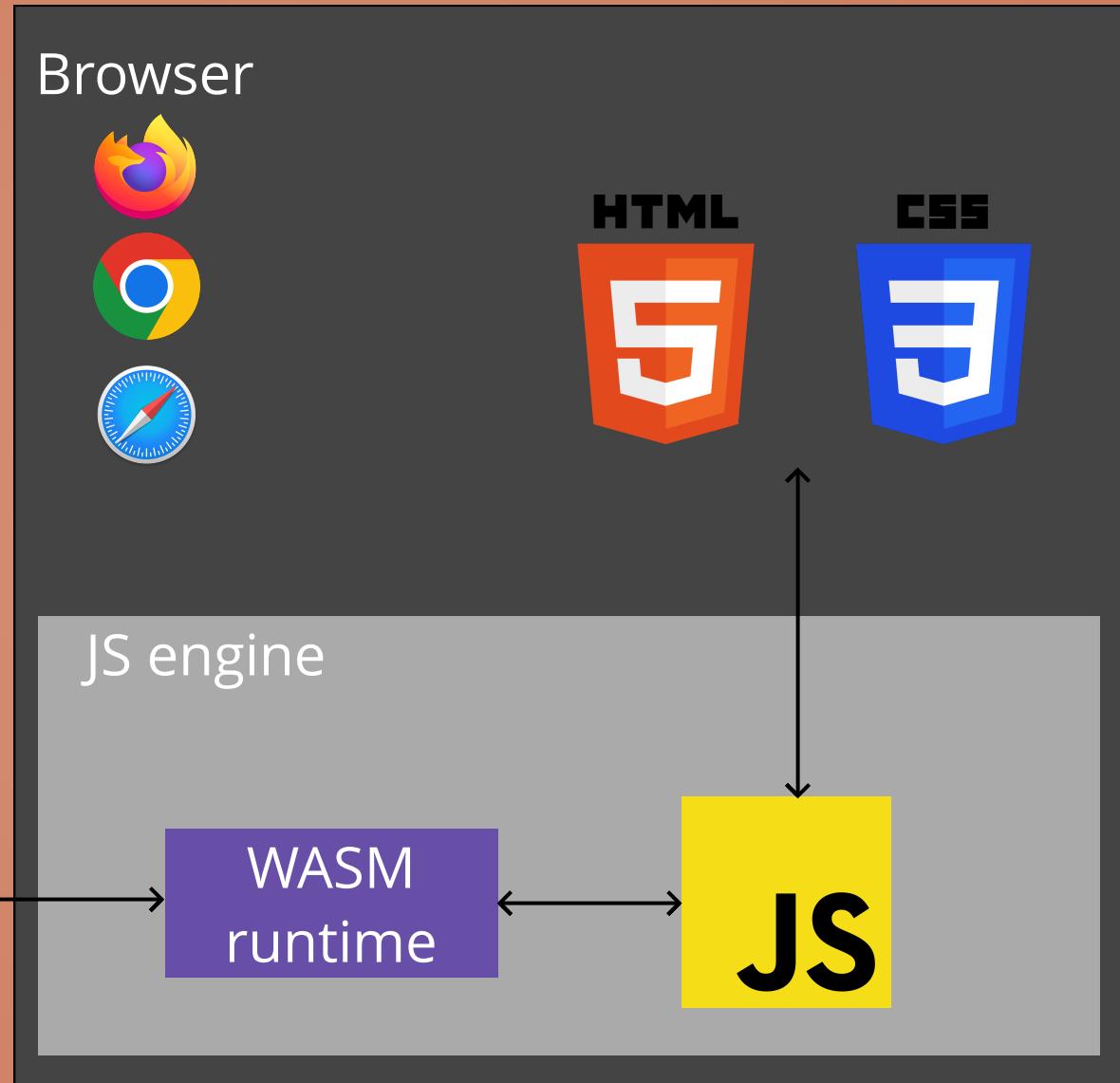
<https://github.com/WebAssembly/wabt>

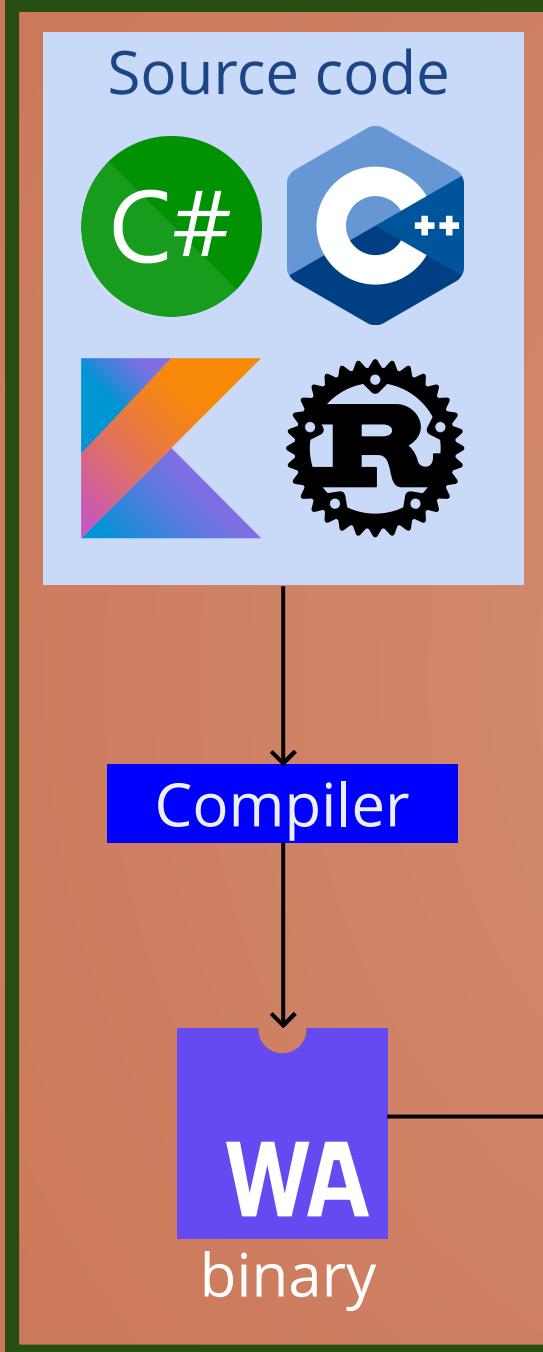


wat format (human readable)

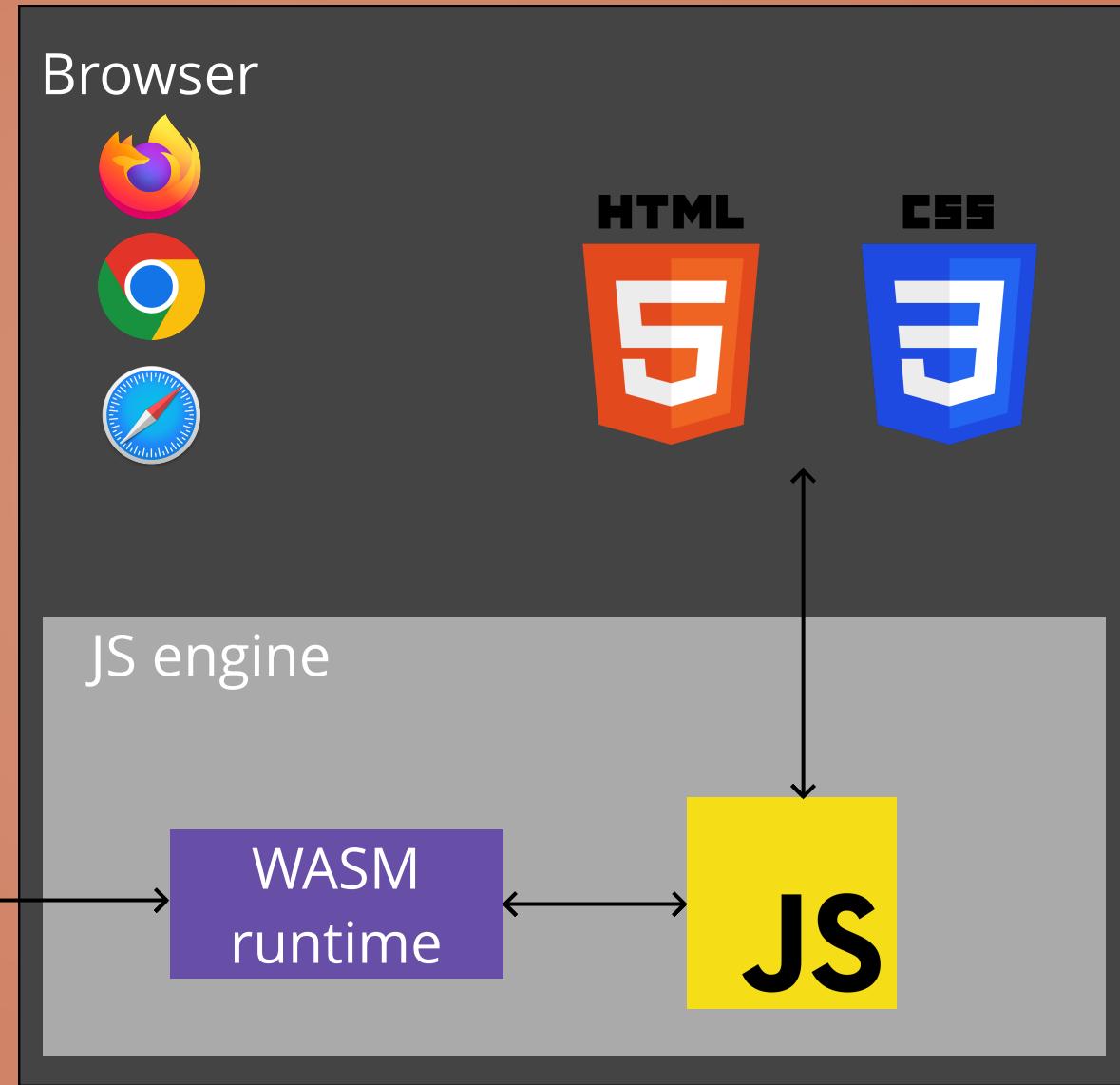
```
1 (module
2   (func (result i32)
3     (i32.const 42)
4   )
5   (export "getUniversalNumber" (func 0))
6 )
```

WASM in browsers



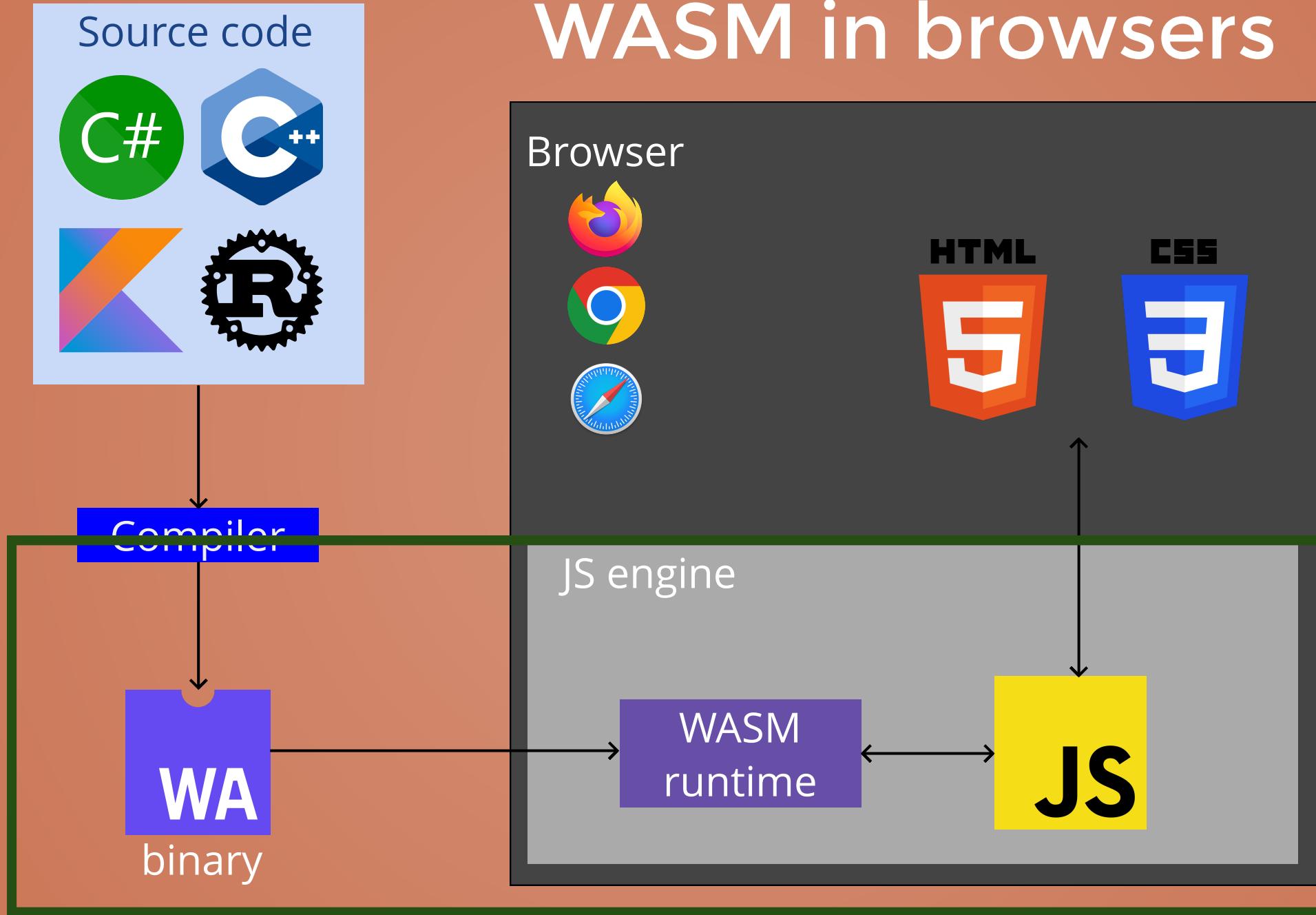


WASM in browsers



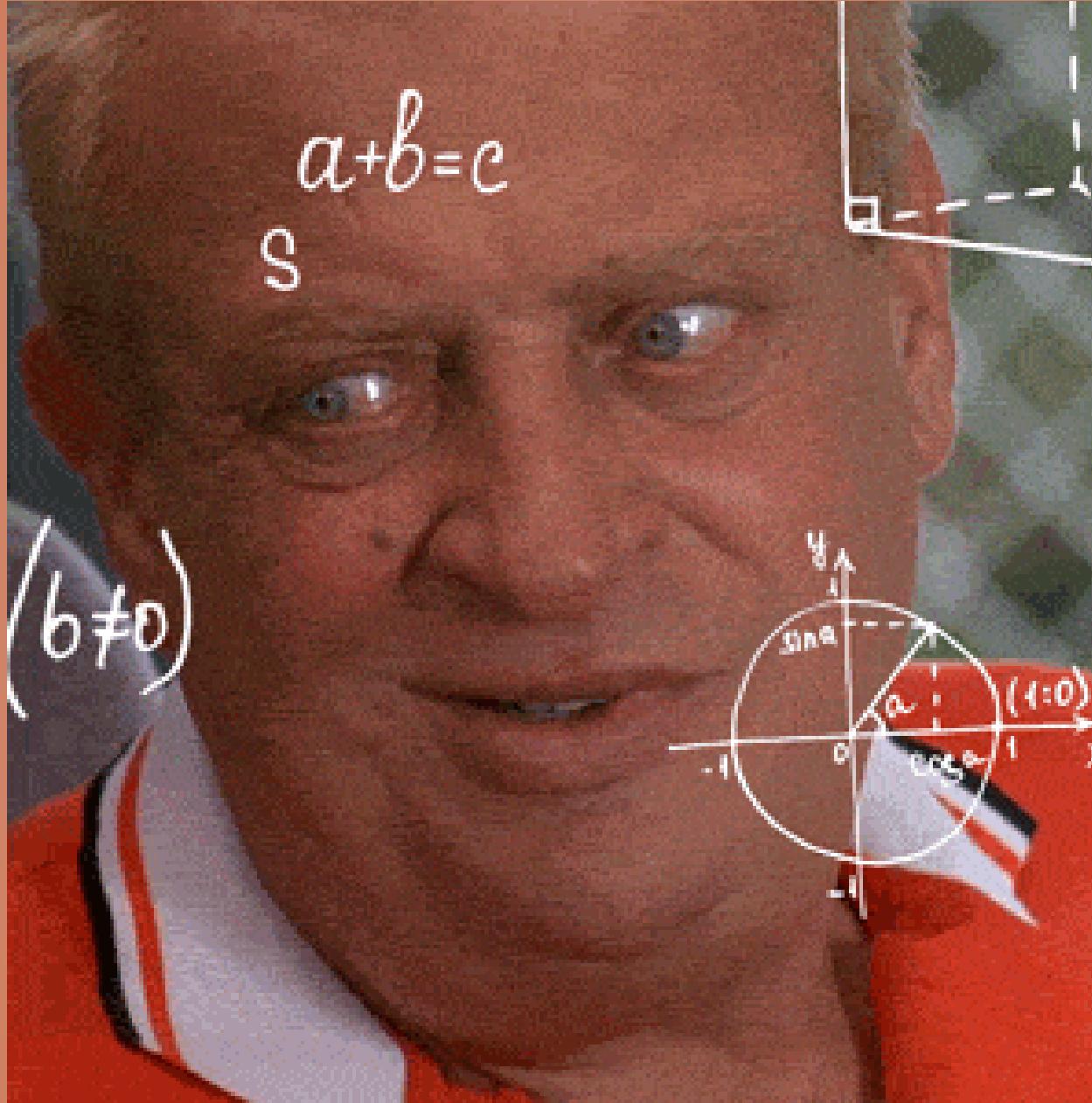
Based on: <https://wasmlabs.dev/articles/docker-without-containers/>

WASM in browsers



WASM on the browser

- A lot of languages compile to WASM
- JS interop



Need to do something related to WASM

CHARTER

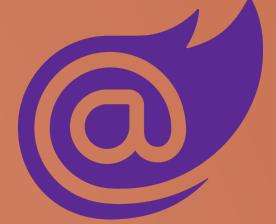
.NET AND WASM



.NET on the browser

- .NET: C# OSS cross-platform framework

.NET on the browser



- .NET: C# OSS cross-platform framework
- Compiles to WASM (Blazor is an example)
- API to interop with JS

.NET 7+ JS Interop

```
1 public partial class MyClass
2 {
3     [JSExport]
4     internal static string Greeting()
5     {
6         var text = $"Hello, {GetHref()}";
7         Console.WriteLine(text);
8         return text;
9     }
10
11    [JSImport("window.location.href", "main.js")]
12    internal static partial string GetHref();
13 }
```



```
1 setModuleImports('main.js', {
2     window: {
3         location: {
4             href: () => globalThis.window.location.href
5         }
6     }
7 });
8
9 const text = exports.MyClass.Greeting();
10 console.log(text);
```



.NET 7+ JS Interop

```
1 public partial class MyClass
2 {
3     [JSExport]
4     internal static string Greeting()
5     {
6         var text = $"Hello, {GetHref()}";
7         Console.WriteLine(text);
8         return text;
9     }
10
11    [JSImport("window.location.href", "main.js")]
12    internal static partial string GetHref();
13 }
```



```
1 setModuleImports('main.js', {
2     window: {
3         location: {
4             href: () => globalThis.window.location.href
5         }
6     }
7 });
8
9 const text = exports.MyClass.Greeting();
10 console.log(text);
```



.NET 7+ JS Interop

```
1 public partial class MyClass
2 {
3     [JSExport]
4     internal static string Greeting()
5     {
6         var text = $"Hello, {GetHref()}";
7         Console.WriteLine(text);
8         return text;
9     }
10
11    [JSImport("window.location.href", "main.js")]
12    internal static partial string GetHref();
13 }
```



```
1 setModuleImports('main.js', {
2     window: {
3         location: {
4             href: () => globalThis.window.location.href
5         }
6     }
7 });
8
9 const text = exports.MyClass.Greeting();
10 console.log(text);
```





Let's port a .NET game !

CHARTERED

MANAGEDDOM

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
- 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://mrglitchesreviews.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

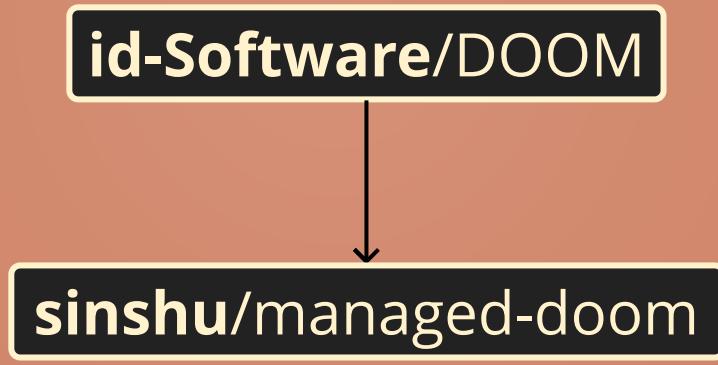
- .NET Port of [LinuxDoom](#) (official source code)
- ManagedDoom V1 Uses [SFML](#) (graphics + audio + input)
 - V2 uses [silk.net](#) (released 27 Dec 2022)
- My work is a fork of **ManagedDoom V1**

id-Software/DOOM



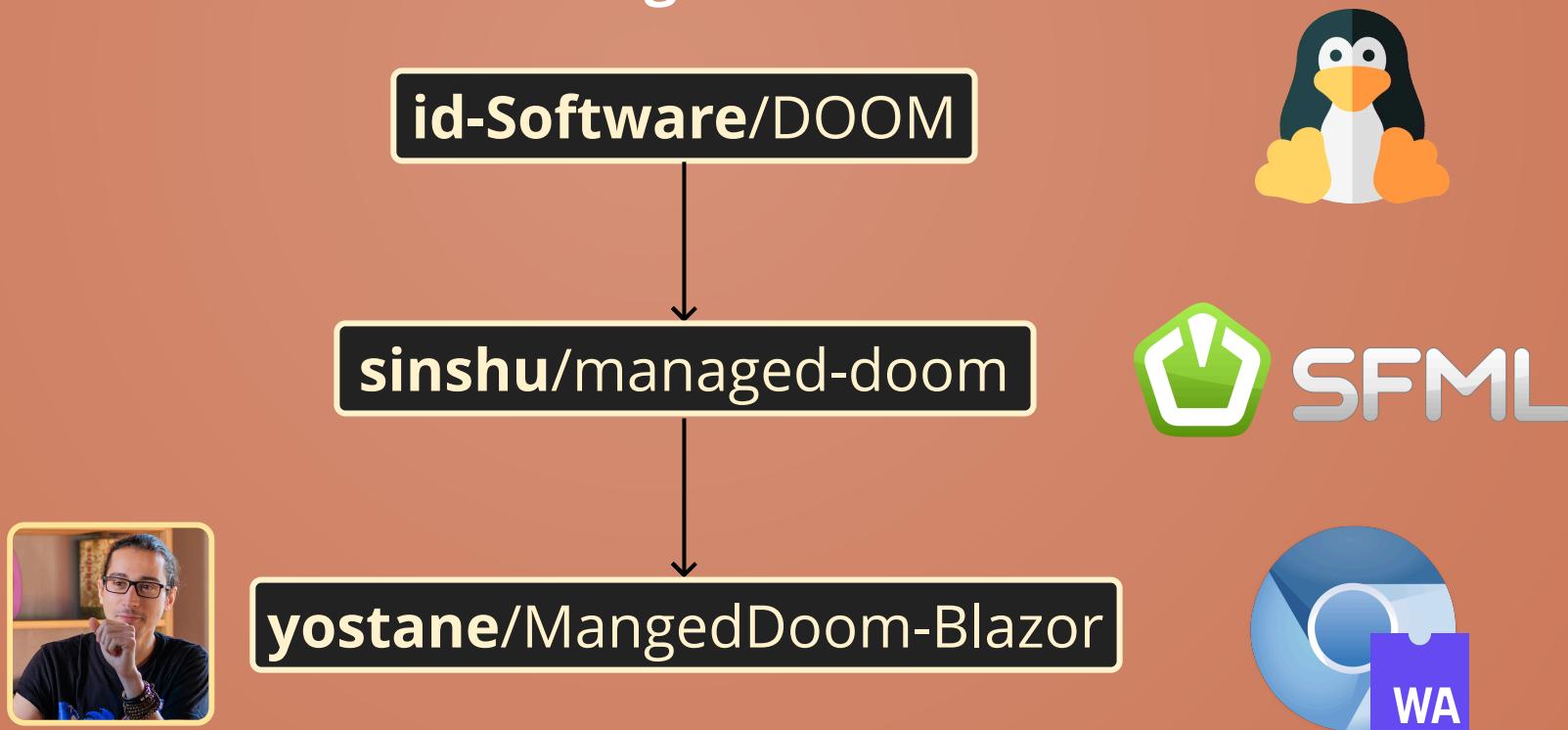
ManagedDoom

- .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**



CHARTERED

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);
```

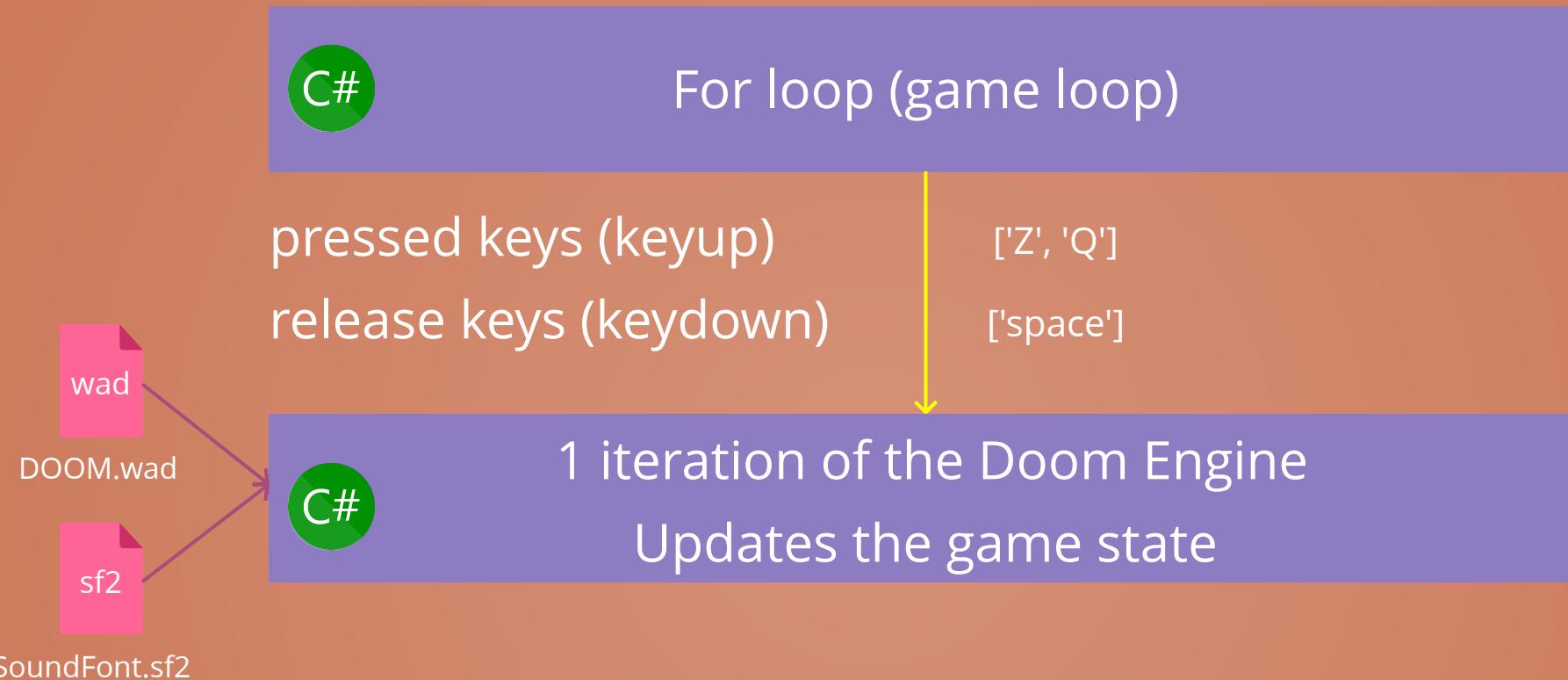
ManagedDoom V1 architecture

ManagedDoom V1 architecture

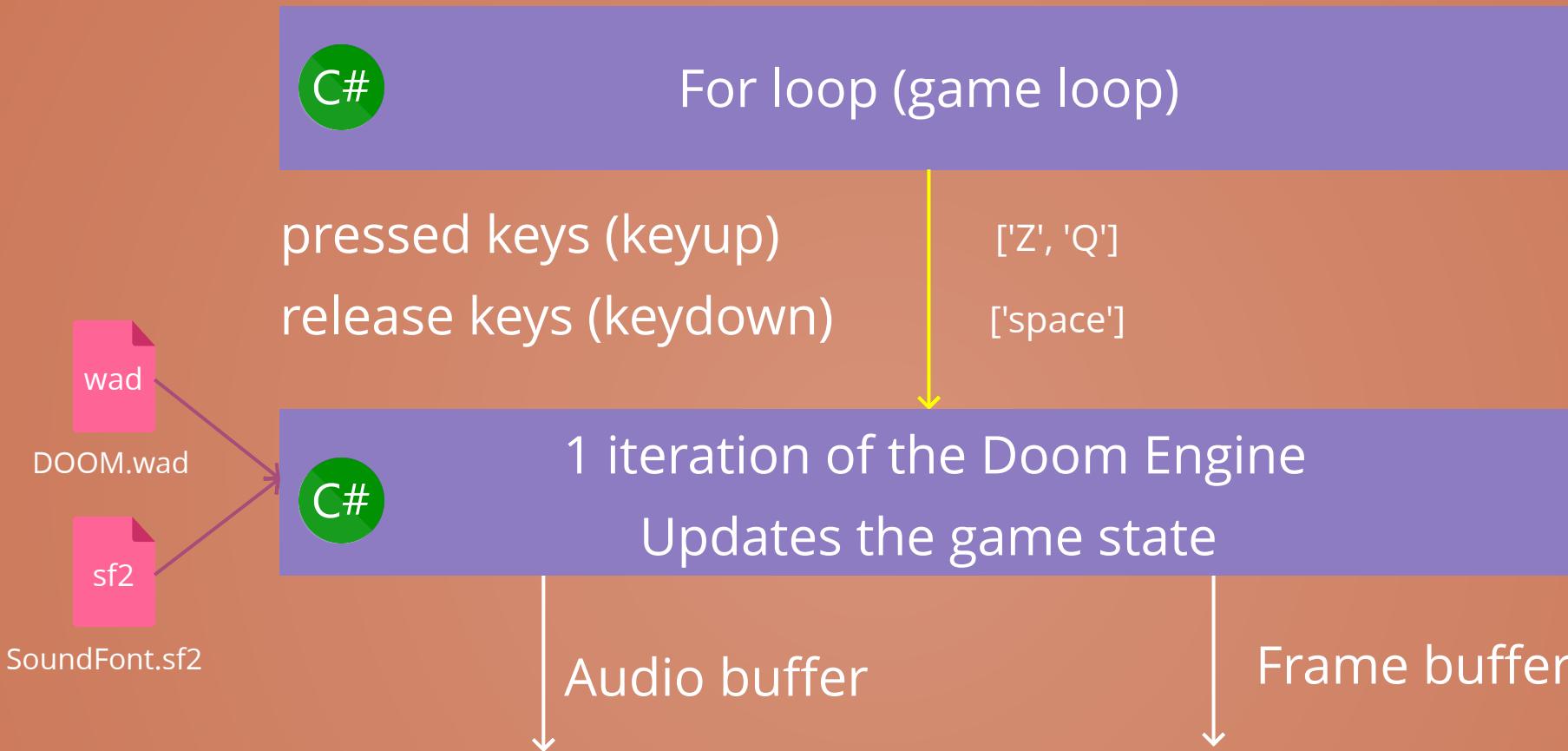
C#

For loop (game loop)

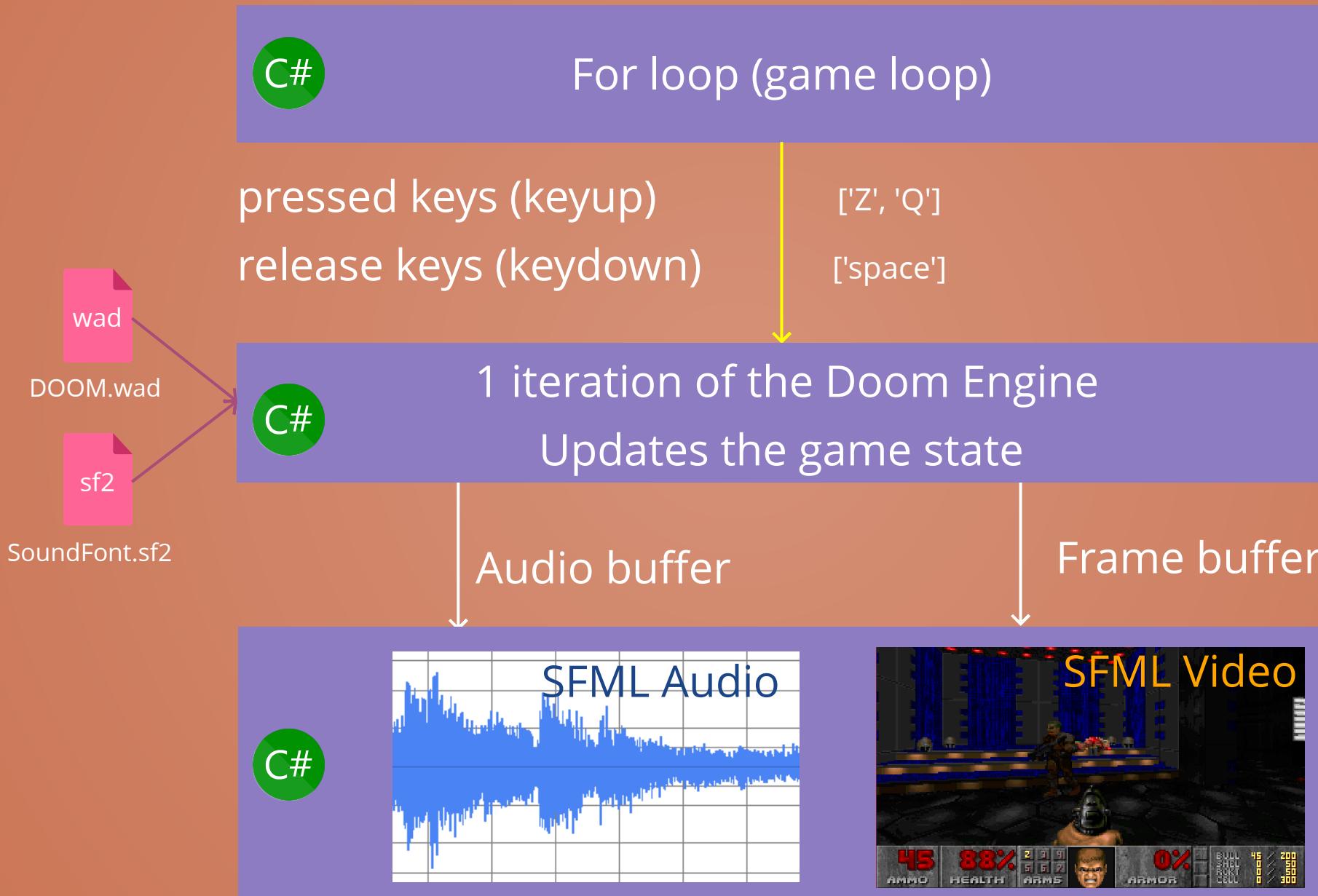
ManagedDoom V1 architecture



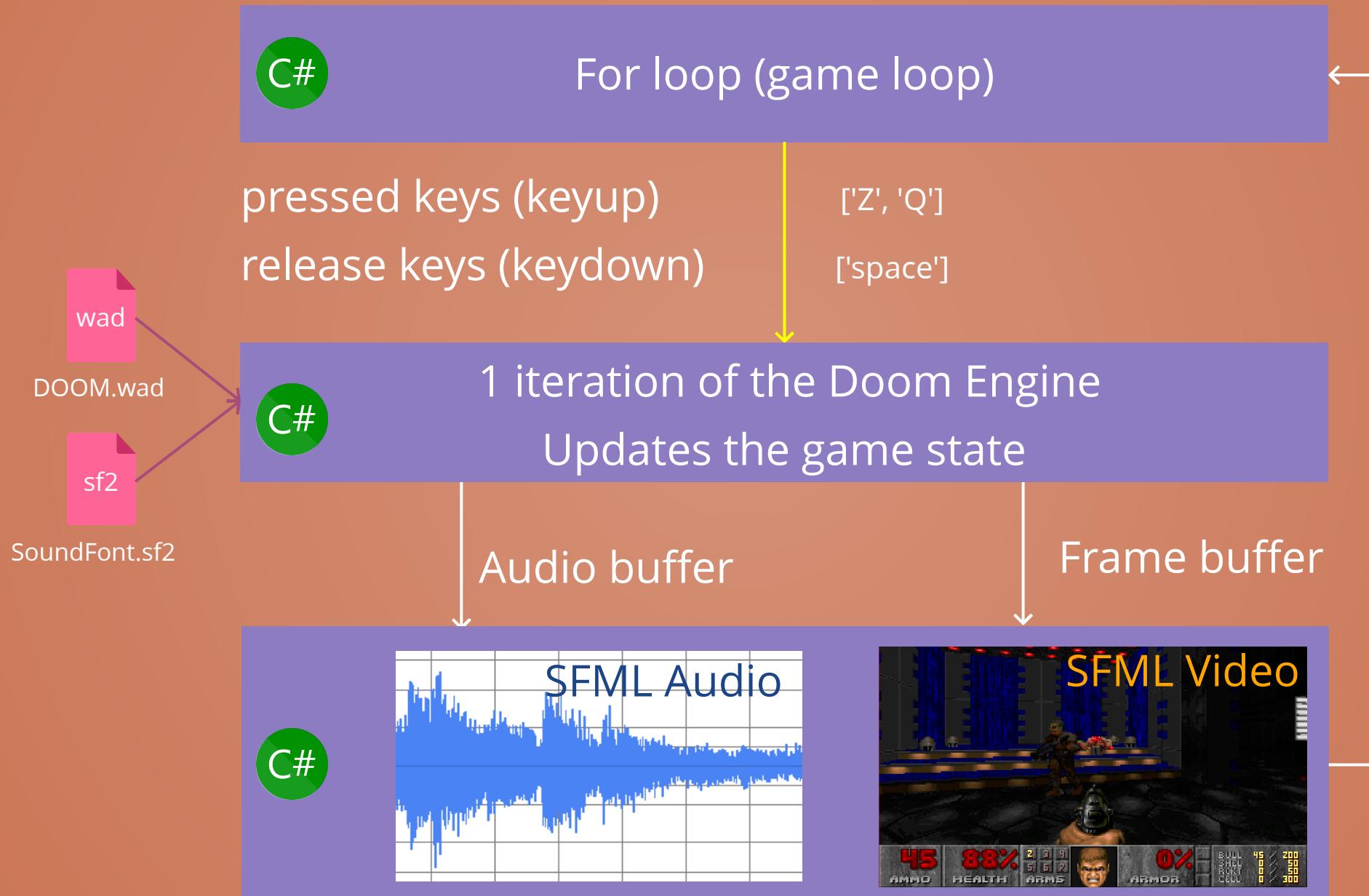
ManagedDoom V1 architecture



ManagedDoom V1 architecture



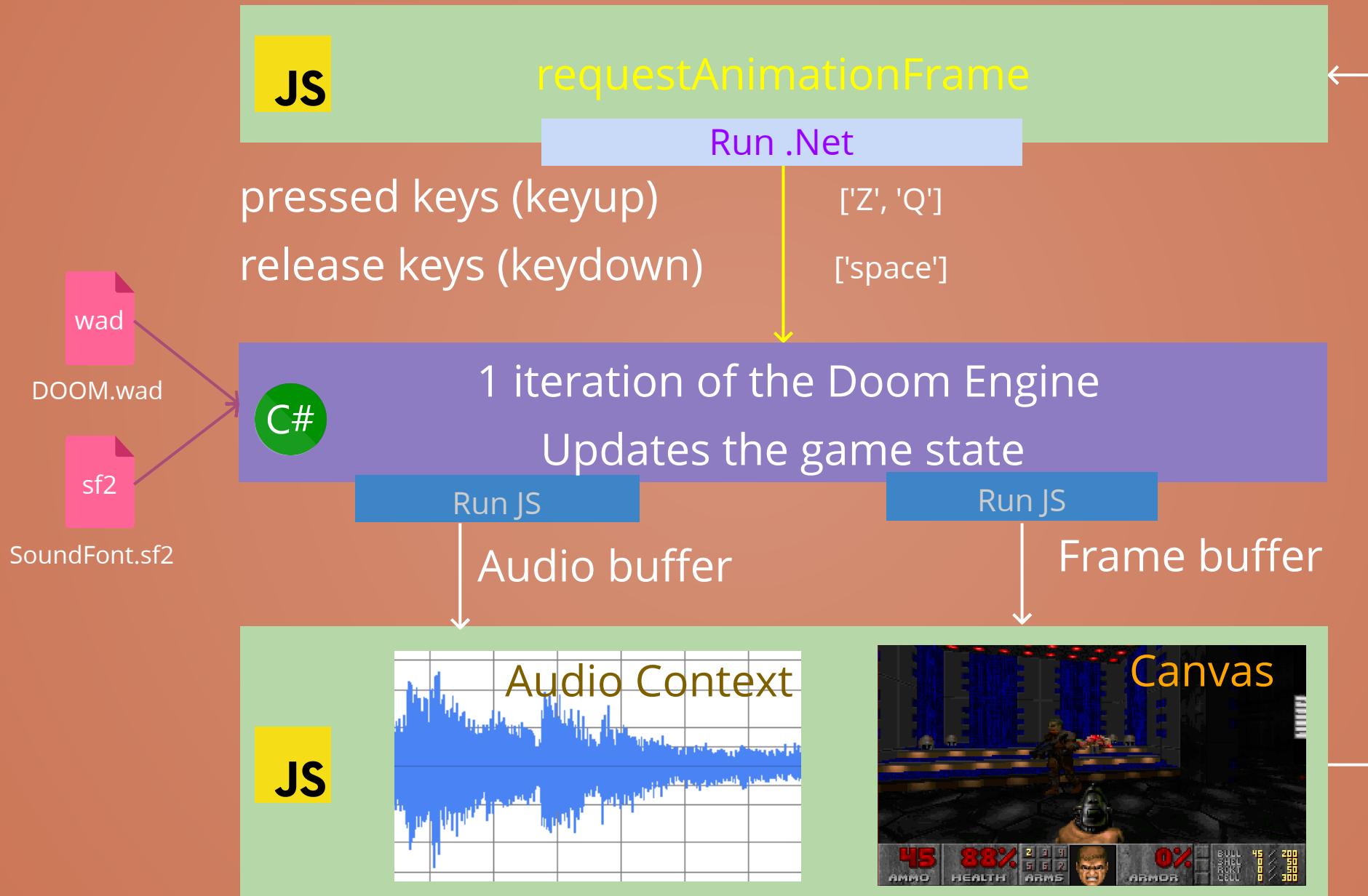
ManagedDoom V1 architecture



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 mark them ("*TODO: implement*")



Porting strategy

- Clone **sinshu/managed-doom** and compile to **.NET WASM**
- Reverse engineer SFML:
 - Put empty implementations where it fails and mark 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 mark 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 }
```

```
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        internal void Dispose()
19        {
20            // TODO: implement
21        }
22    }
23 }
```

CHARTER 4

KNEE-DEEP IN THE CODE

CHAPTER 4

KNEE-DEEP IN THE CODE

Entry point



```
1 <html>
2   <head>
3     <!-- Sets .Net interop and starts the game loop -->
4     <script type="module" src=".main.js"></script>
5   </head>
6   <body>
7     <canvas id="canvas" width="320" height="200"
8           style="image-rendering: pixelated" />
9   </body>
10 </html>
```

Entry point



```
1 <html>
2   <head>
3     <!-- Sets .Net interop and starts the game loop -->
4     <script type="module" src=".main.js"></script>
5   </head>
6   <body>
7     <canvas id="canvas" width="320" height="200"
8           style="image-rendering: pixelated" />
9   </body>
10 </html>
```

Entry point



```
1 <html>
2   <head>
3     <!-- Sets .Net interop and starts the game loop -->
4     <script type="module" src="./main.js"></script>
5   </head>
6   <body>
7     <canvas id="canvas" width="320" height="200"
8           style="image-rendering: pixelated" />
9   </body>
10 </html>
```

Running the gameloop

main.js

```
1 import { dotnet } from "./dotnet.js";
2 const { getAssemblyExports, getConfig } = await dotnet.create();
3 const exports = await getAssemblyExports(getConfig()).mainAssemblyName;
4 await dotnet.run();
5
6 function gameLoop(timestamp) {
7     if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8         lastFrameTimestamp = timestamp;
9         exports.BlazorDoom.MainJS.UpdateGameState(keys);
10    }
11    requestAnimationFrame(gameLoop);
12 }
```



Running the gameloop

main.js

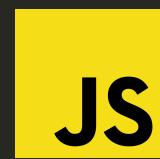
```
1 import { dotnet } from "./dotnet.js";
2 const { getAssemblyExports, getConfig } = await dotnet.create();
3 const exports = await getAssemblyExports(getConfig()).mainAssemblyName;
4 await dotnet.run();
5
6 function gameLoop(timestamp) {
7     if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8         lastFrameTimestamp = timestamp;
9         exports.BlazorDoom.MainJS.UpdateGameState(keys);
10    }
11    requestAnimationFrame(gameLoop);
12 }
```



Running the gameloop

main.js

```
1 import { dotnet } from "./dotnet.js";
2 const { getAssemblyExports, getConfig } = await dotnet.create();
3 const exports = await getAssemblyExports(getConfig()).mainAssemblyName;
4 await dotnet.run();
5
6 function gameLoop(timestamp) {
7     if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8         lastFrameTimestamp = timestamp;
9         exports.BlazorDoom.MainJS.UpdateGameState(keys);
10    }
11    requestAnimationFrame(gameLoop);
12 }
```



Running the gameloop

main.js

```
1 import { dotnet } from "./dotnet.js";
2 const { getAssemblyExports, getConfig } = await dotnet.create();
3 const exports = await getAssemblyExports(getConfig()).mainAssemblyName;
4 await dotnet.run();
5
6 function gameLoop(timestamp) {
7     if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8         lastFrameTimestamp = timestamp;
9         exports.BlazorDoom.MainJS.UpdateGameState(keys);
10    }
11    requestAnimationFrame(gameLoop);
12 }
```



Running the gameloop

main.js

```
1 import { dotnet } from "./dotnet.js";
2 const { getAssemblyExports, getConfig } = await dotnet.create();
3 const exports = await getAssemblyExports(getConfig()).mainAssemblyName;
4 await dotnet.run();
5
6 function gameLoop(timestamp) {
7     if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8         lastFrameTimestamp = timestamp;
9         exports.BlazorDoom.MainJS.UpdateGameState(keys);
10    }
11    requestAnimationFrame(gameLoop);
12 }
```



Running the gameloop

main.js

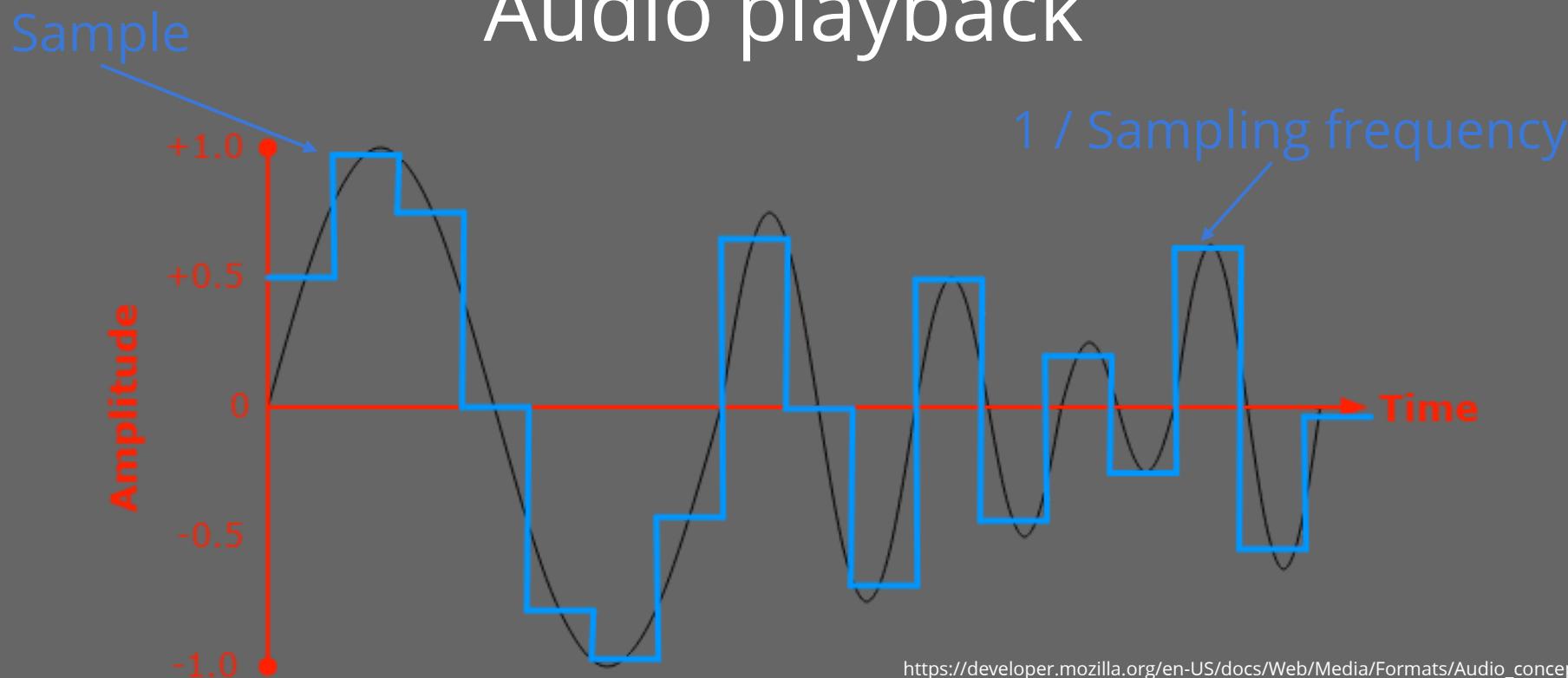
```
1 import { dotnet } from "./dotnet.js";
2 const { getAssemblyExports, getConfig } = await dotnet.create();
3 const exports = await getAssemblyExports(getConfig()).mainAssemblyName;
4 await dotnet.run();
5
6 function gameLoop(timestamp) {
7     if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8         lastFrameTimestamp = timestamp;
9         exports.BlazorDoom.MainJS.UpdateGameState(keys);
10    }
11    requestAnimationFrame(gameLoop);
12 }
```



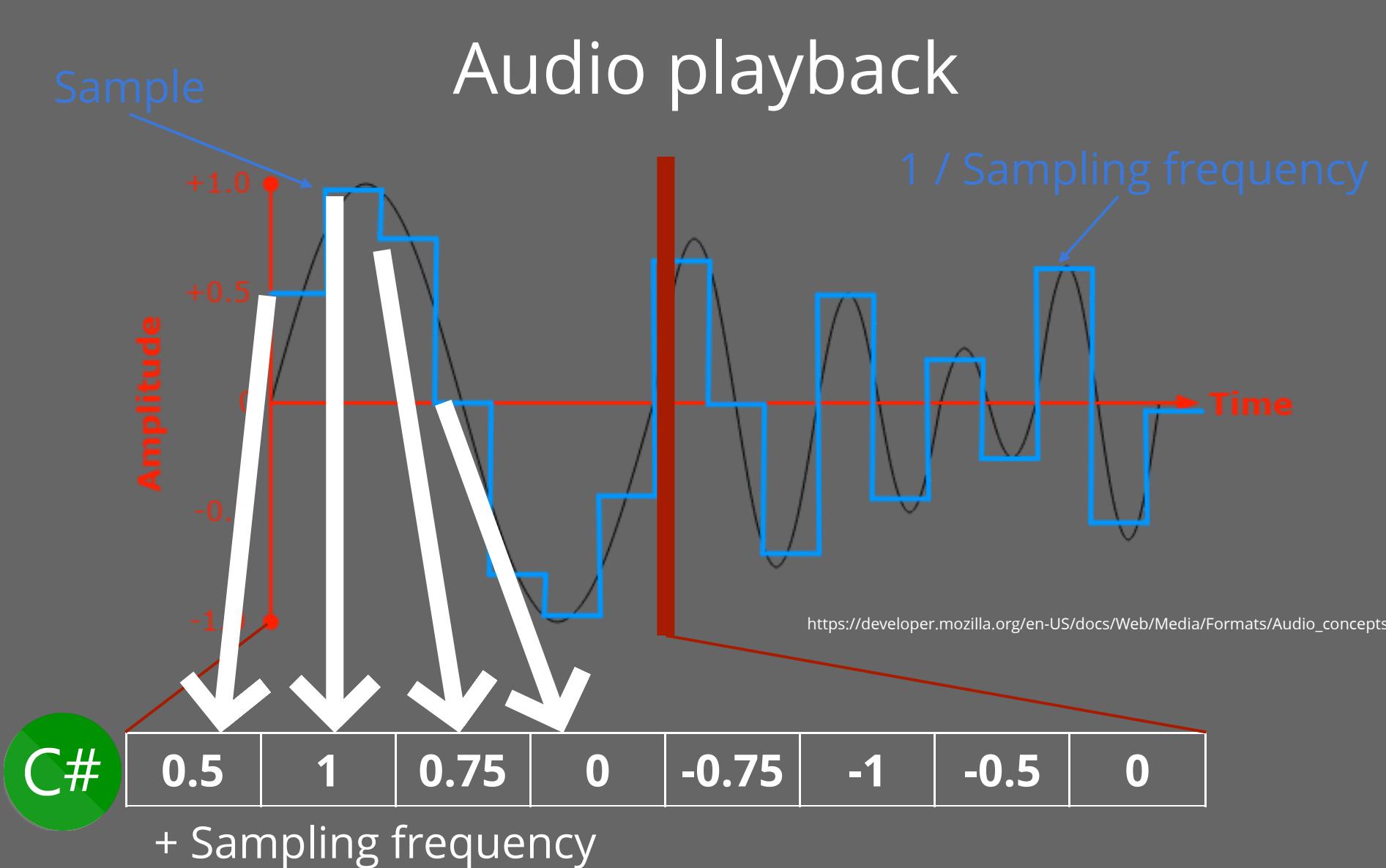
```
1 public partial class MainJS // this name is required
2 {
3     public static void Main()
4     {
5         app = new ManagedDoom.DoomApplication();
6     }
7
8     [JSExport] // Can be imported from JS
9     public static void UpdateGameState(int[] keys)
10    { // computes the next frame and sounds
11        managedDoom.UpdateGameState(keys);
12    }
13 }
```



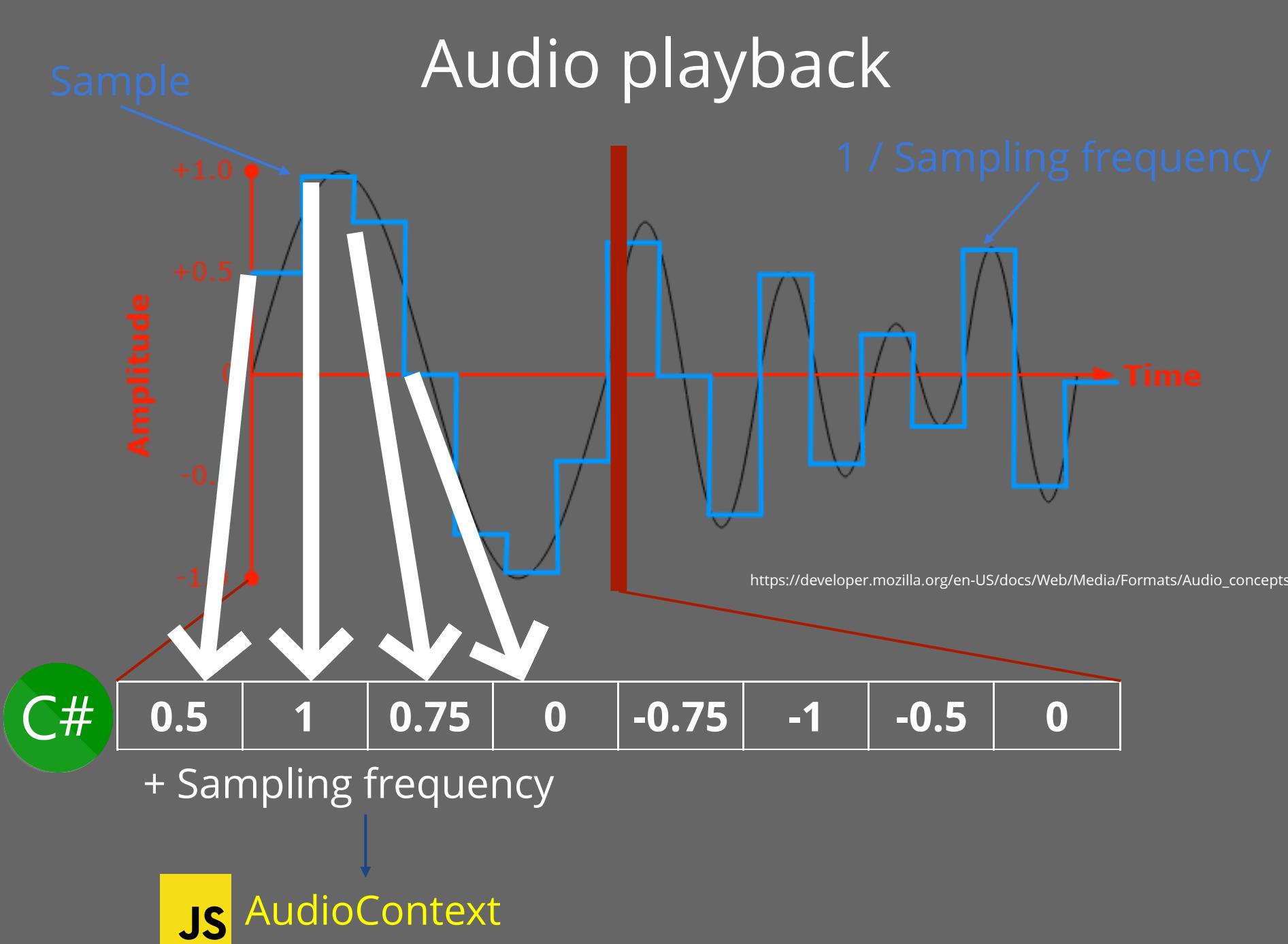
Audio playback



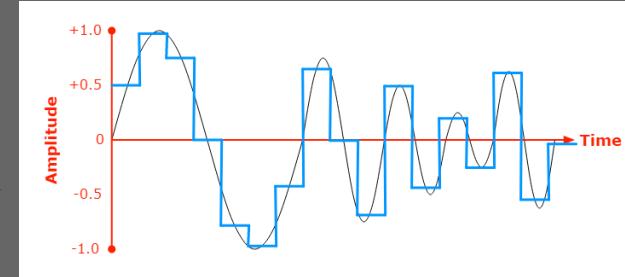
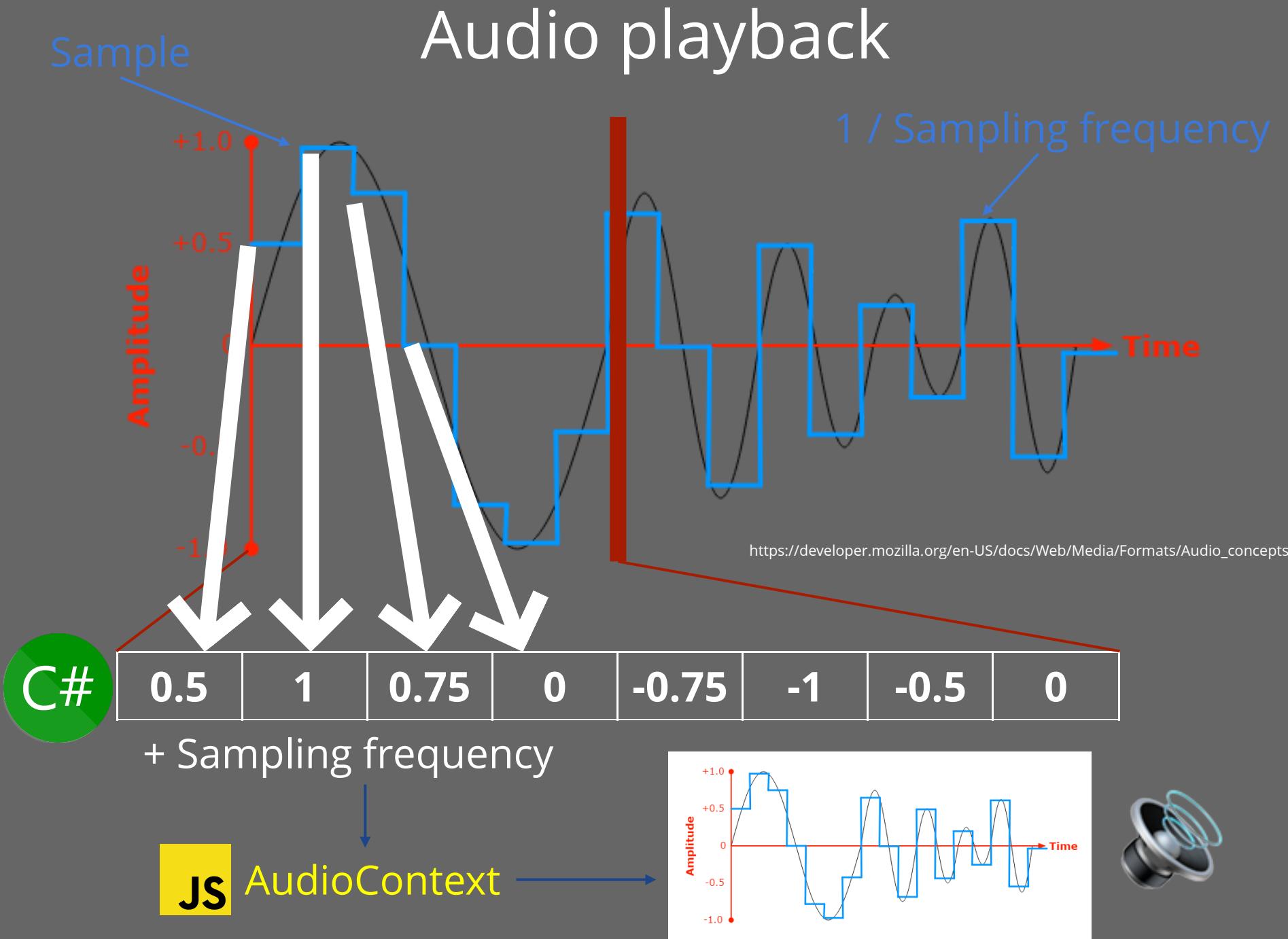
Audio playback



Audio playback



Audio playback



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 }
```



```
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 }
```



Audio playback

JS

```
1 export function playSound(samples, sampleRate) {
2     audioContext = new AudioContext({
3         sampleRate: sampleRate,
4     });
5     const length = samples.length;
6     const audioBuffer = audioContext.createBuffer(
7         1,
8         length,
9         sampleRate
10    );
11
12     var channelData = audioBuffer.getChannelData(0);
13     for (let i = 0; i < length; i++) {
14         // noralize the sample to be between -1 and 1
15         channelData[i] = samples[i] / 0xffff;
16     }
17
18     var source = audioContext.createBufferSource();
19     source.buffer = audioBuffer;
20     source.connect(audioContext.destination);
21     source.start();
22 }
```

Audio playback

JS

```
1 export function playSound(samples, sampleRate) {
2     audioContext = new AudioContext({
3         sampleRate: sampleRate,
4     });
5     const length = samples.length;
6     const audioBuffer = audioContext.createBuffer(
7         1,
8         length,
9         sampleRate
10    );
11
12     var channelData = audioBuffer.getChannelData(0);
13     for (let i = 0; i < length; i++) {
14         // noralize the sample to be between -1 and 1
15         channelData[i] = samples[i] / 0xffff;
16     }
17
18     var source = audioContext.createBufferSource();
19     source.buffer = audioBuffer;
20     source.connect(audioContext.destination);
21     source.start();
22 }
```

Audio playback

JS

```
1 export function playSound(samples, sampleRate) {
2     audioContext = new AudioContext({
3         sampleRate: sampleRate,
4     });
5     const length = samples.length;
6     const audioBuffer = audioContext.createBuffer(
7         1,
8         length,
9         sampleRate
10    );
11
12     var channelData = audioBuffer.getChannelData(0);
13     for (let i = 0; i < length; i++) {
14         // noralize the sample to be between -1 and 1
15         channelData[i] = samples[i] / 0xffff;
16     }
17
18     var source = audioContext.createBufferSource();
19     source.buffer = audioBuffer;
20     source.connect(audioContext.destination);
21     source.start();
22 }
```

From 1D frame to a 2D frame



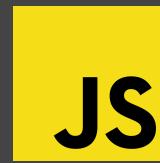
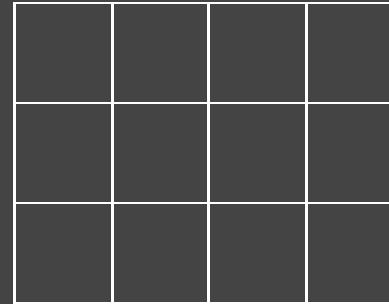
Frame data
12 bytes

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

Color palette



0	1	2	3
Red square	Green square	Purple square	Yellow square



From 1D frame to a 2D frame



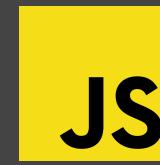
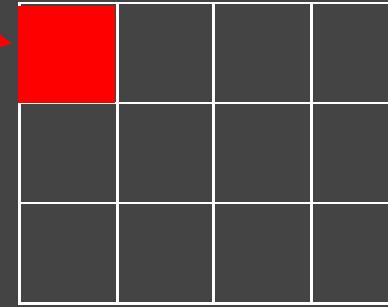
Frame data
12 bytes

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

Color palette



0	1	2	3
Red	Green	Purple	Yellow



From 1D frame to a 2D frame



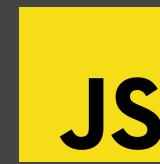
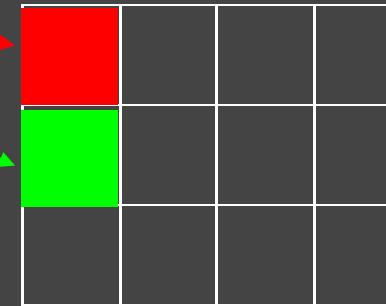
Frame data
12 bytes

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

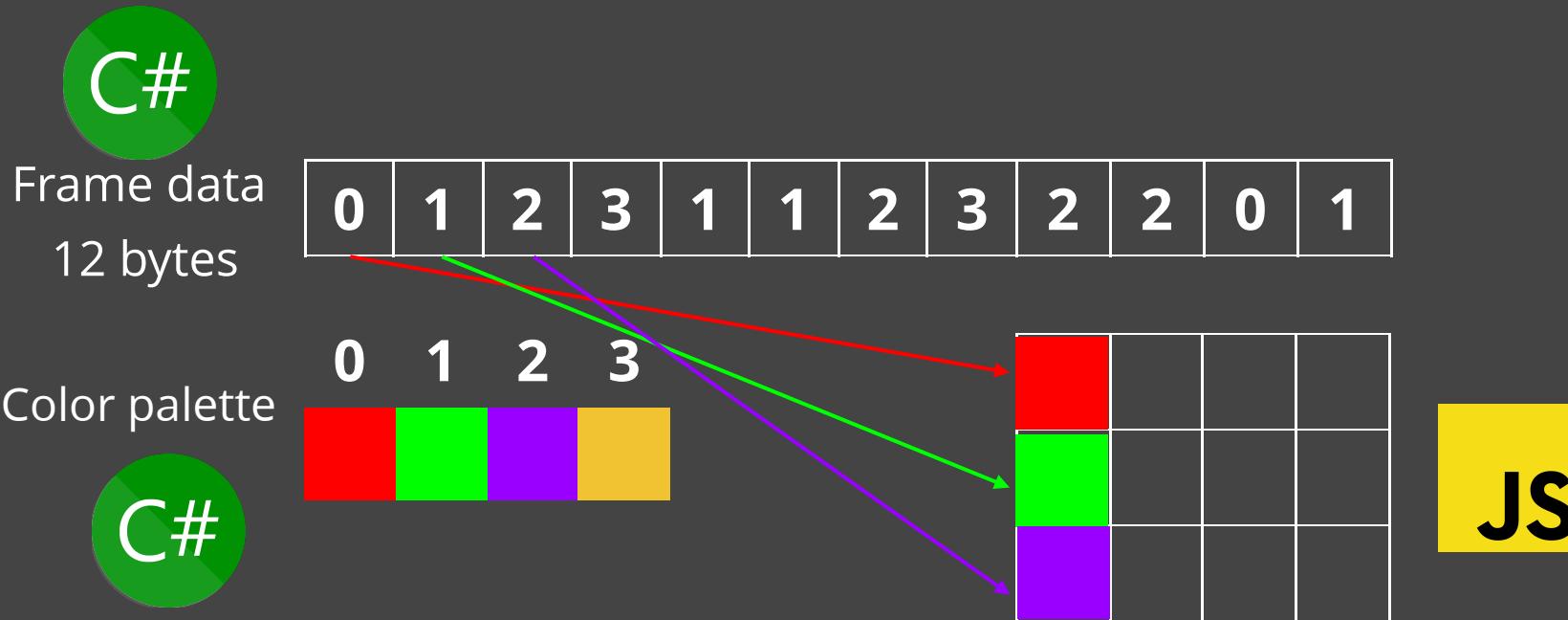
Color palette



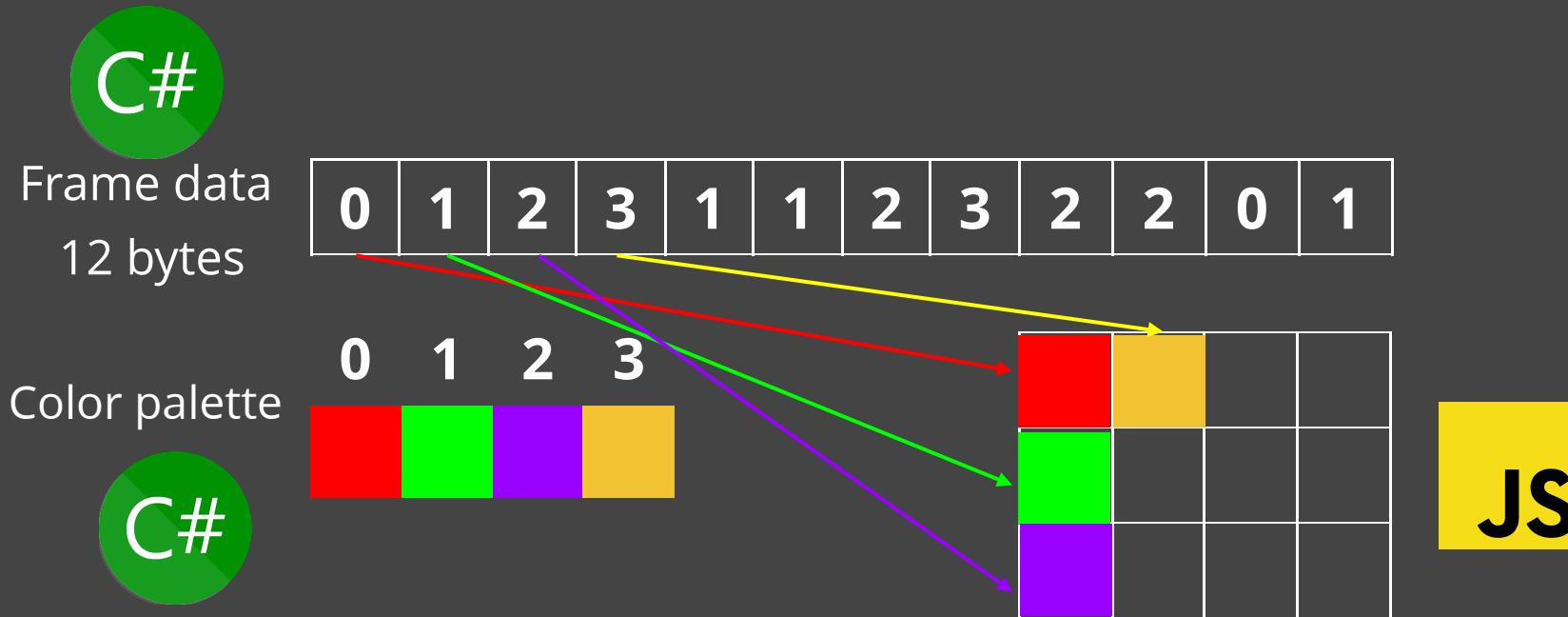
0	1	2	3
Red	Green	Purple	Yellow



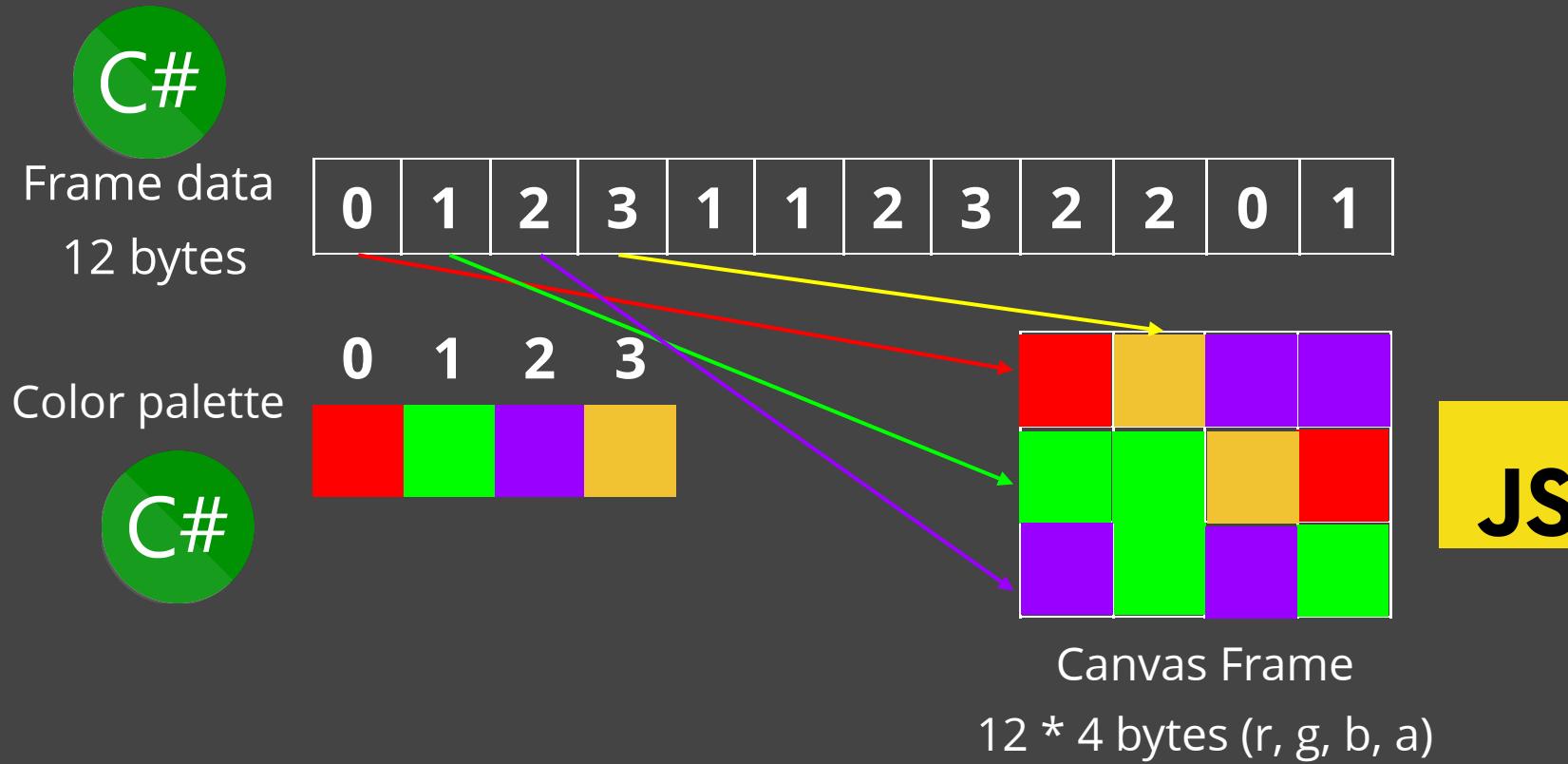
From 1D frame to a 2D frame



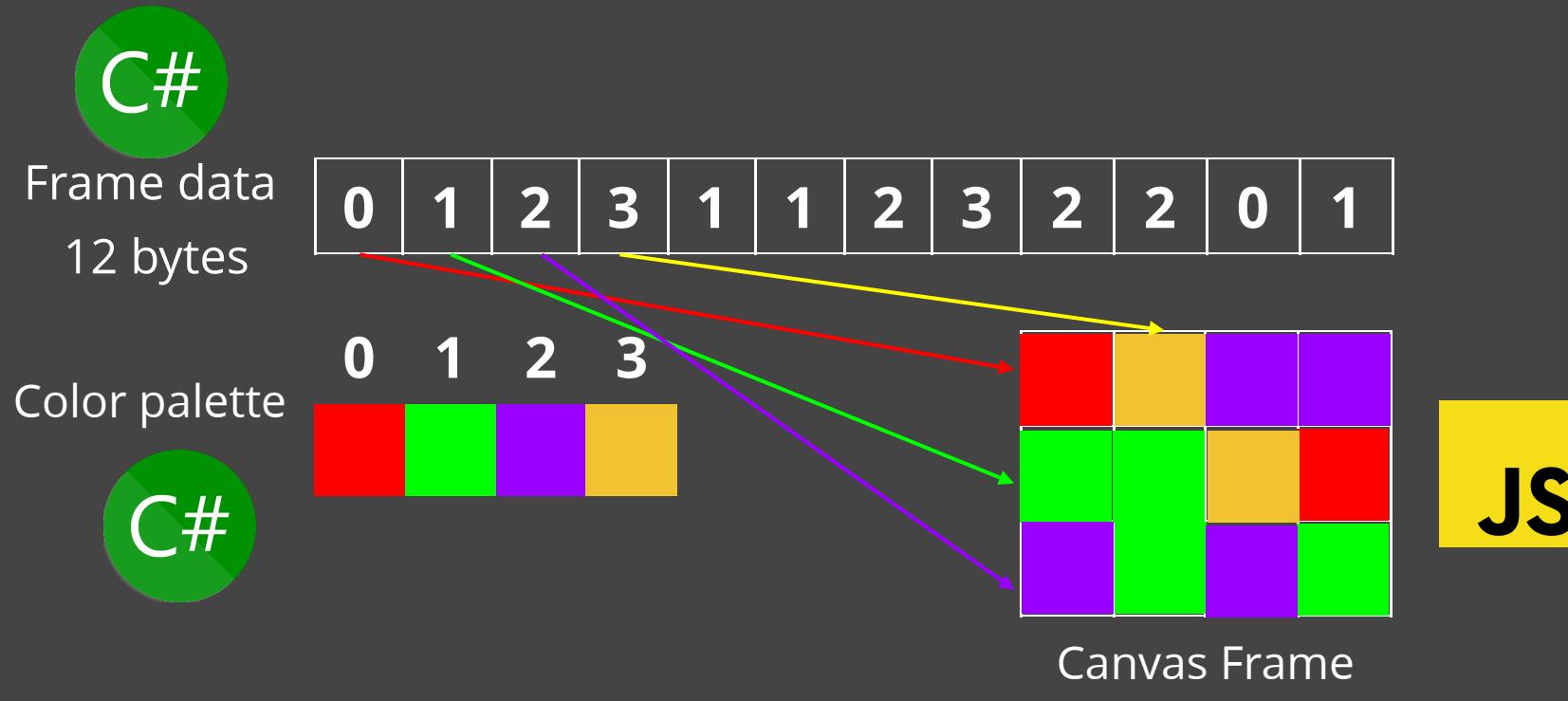
From 1D frame to a 2D frame



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 }
```



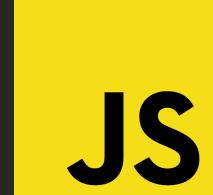
```
1 namespace BlazorDoom
2 {
3     [SupportedOSPlatform("browser")]
4     public partial class Renderer
5     {
6         [JSImport("drawOnCanvas", "blazorDoom/renderer.js")]
7         internal static partial string renderOnJS(byte[] screenData,
8                                         int[] colors);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



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



Rendering a Frame buffer

 JS

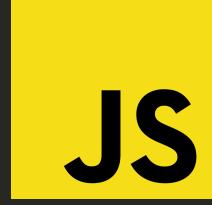
```
1 export function drawOnCanvas(screenData, colors) {
2     const context = getCanvas().context;
3     const imageData = context.createImageData(320, 200);
4     let y = 0;
5     let x = 0;
6     for (let i = 0; i < screenData.length; i += 1) {
7         const dataIndex = (y * width + x) * 4;
8         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 }
```

Rendering a Frame buffer

 JS

```
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 }
```

Rendering a Frame buffer

 JS

```
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 }
```

CHARTER 5

DEMO(ED)



GLOBAL PARTNERS

**NEXT STEPS AND
CONCLUSION**

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**

SIDE
DECK



SOURCE
CODE



THANKS FOR PLAYING !
ANY QUESTION ?



Yassine
Benabbas
  
@yostane

Please leave a review



A screenshot of a mobile application interface for a conference or event. At the top, there is a horizontal timeline with orange circles containing the dates: 7 OCT., 8 OCT., 9 OCT., 10 OCT., and 11 OCT. Below the timeline, a dark grey bar displays the time '16:50 > 17:20' and a badge indicating 'overlaps 1 slot'. The main content area features a session card. The card has a green 'Mind the Geek' tag. The title of the session is 'Run your favorite games everywhere with WASM: the BlazorDoom use case'. Below the title, the speaker is listed as 'Yassine Benabbas (Worldline)'. To the right of the speaker's name is a small circular profile picture of a man with glasses. In the bottom right corner of the card, there is a bookmark icon with the number '49' next to it. The background of the app interface is light grey, and there is a small upward arrow icon in the top right corner.

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>