

# Visualizzatore 3D Doors (Blazor)

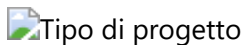
---

- Visualizzatore 3D Doors (Blazor)
  - [\[0\] Prerequisiti](#)
  - [\[1\] Creare un'applicazione Web \(Blazor server app\)](#)
  - [⚠ IMPORTANTE PER IL CORRETTO FUNZIONAMENTO DEL VISUALIZZATORE](#)
  - [\[2\] Setup helper per la comunicazione](#)
  - [\[3\] Aggiungere i riferimenti ai file .js](#)
  - [\[4\] Setup componente blazor per il render](#)
  - [\[5\] Setup della chiamata alla funzione di render](#)

## [0] Prerequisiti

- .NET SDK installato

## [1] Creare un'applicazione Web (Blazor server app)



Tipo di progetto

Una volta creata l'applicazione web creare nella cartella **wwroot** una sottocartella denominandola **lib**. Se si importa localmente inl pacchetto npm viene creata alla fase di definizione del path di destinazione



Import NPM

E' possibile importare il pacchetto npm che si trova sul nostro repository nexus (<https://nexus.steamware.net/#browse/browse:npm-hosted:webgl-door-visualizer>) con il comando seguente

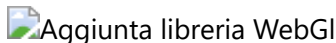
```
npm install webgl-door-visualizer@1.2.240622.1216
```

Dove il numero versione va modificato di conseguenza a quanto voluto.



Creazione cartella lib

Successivamente copiare l'intera cartella **src** fornita ed incollarla nella cartella **lib** precedentemente creata.



Aggiunta libreria WebGL

## ⚠ IMPORTANTE PER IL CORRETTO FUNZIONAMENTO DEL VISUALIZZATORE

Visual studio riconosce e permette l'utilizzo di alcuni tipi di file. L'estensione **.3dm** non rientra nella categoria perciò è importante ricordarsi di includere quest'ultima nella classe **Program.cs** le seguenti righe di codice:

```
var provider = new
Microsoft.AspNetCore.StaticFiles.FileExtensionContentTypeProvider();
provider.Mappings[".3dm"] = "model"; // aggiungere l'estensione desiderata e il
suo tipo

app.UseStaticFiles(new StaticFileOptions
```

```
{
    ContentTypeProvider = provider
});
```

inoltre ma aggiunta la configurazione per distribuire file 3dm dalla cartella appositamente prevista (ed esterna al programma epr "sopravvivere" agli update dello stesso)

```
string path3dm = configuration.GetValue<string>("ServerConf:path3dm") ??
configuration.GetValue<string>("OptConf:path3dm") ?? "";
if (!string.IsNullOrEmpty(path3dm))
{
    // verifico esista folder disegni
    if (Directory.Exists(path3dm))
    {
        // gestione cartella x PDF
        app.UseStaticFiles(new StaticFileOptions
        {
            FileProvider = new PhysicalFileProvider(path3dm),
            RequestPath = "/3dm",
            ContentTypeProvider = provider
        });
    }
}
```

## 2] Setup helper per la comunicazione

Nella cartella **lib** dovrebbe trovarsi il file `draw_call.js`. Questo sarà il file che farà da tramite tra chi gestisce il visualizzatore ( `WebGl` ) e chi mostrerà il visualizzatore ( `WebGlViewer` )

Il file conterrà le seguenti righe di codice :

```
import { webgl_original } from './webgl_draw.js';

const WGL = new webgl_original();

const infoDiv = document.getElementById( "infoDiv" ) ;
const doorDiv = document.getElementById( "DoorRender" ) ;
const cubeDiv = document.getElementById( "Ref1Render" ) ;
infoDiv.style.display = "none";

var isAnimated = true;
var isRotating = true;
var showQuote = true;
var showGrid = true;

let options ={
    modelPath: "https://iis01.egalware.com/Test3D/THREEJS_DOORS/Door_models",
    fName: getUrlParameter("src")
}
```

```
// init controllo
WGL.initcall(options);
```

In questo caso sono presenti due funzioni :

- **initcall** permette di avviare i calcoli che permettono il render del visualizzatore 3D. I parametri richiesti :
  - **modelPath**: path da dove vengono restituiti i file 3dm (da configurare nel program.cs come sopra);
  - **fName**: nome del file ( con estensione **.3dm** ) del quale effettuare il render.

**N.B.:** E' fondamentale caricare i propri modelli da visualizzare nella cartella sopra definita in program.cs.

### 3 Aggiungere i riferimenti ai file .js

Per far si che il codice scritto nel file di base della libreria e nel file **helper** è importante aggiungere le seguenti righe alla pagina

Pages/\_layout.cshtml :

```
<script src="~/lib/node_modules/webgl-door-visualizer/webgl_draw.js"
type="module"></script>
<script src="~/lib/node_modules/webgl-door-visualizer/draw_call.js"></script>
```

Di seguito il risultato di come si presenterà la pagina :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <base href="/" />
  <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
  <link href="css/site.css" rel="stylesheet" />
  <link href="test.styles.css" rel="stylesheet" />
  <component type="typeof(HeadOutlet)" render-mode="ServerPrerendered" />
</head>
<body>
  @RenderBody()

  <div id="blazor-error-ui">
    <environment include="Staging,Production">
      An error has occurred. This application may no longer respond until
      reloaded.
    </environment>
```

```

        <environment include="Development">
            An unhandled exception has occurred. See browser dev tools for
            details.
        </environment>
        <a href="" class="reload">Reload</a>
        <a class="dismiss">✕</a>
    </div>

    <script src="~/lib/WebGl/draw/webgl_draw.js" type="module"></script>
    <script src="~/lib/WebGl/draw/webgl_helper.js" type="module"></script>

    <script src="_framework/blazor.server.js"></script>
</body>
</html>

```

## 4 Setup componente blazor per il render

Successivamente creiamo un nuovo componente razor nella cartelle delle pagine `Pages/WebGlViewer`.

Nella nuova pagina inserire le seguenti righe di codice che permetteranno di effettuare il render del visualizzatore.

```

@inject IJSRuntime JSRuntime
@page "/WebGlViewer"

<div class="d-flex justify-content-between bg-dark bg-opacity-75">
    <div class="p-2">
        <div class="btn-group" role="group" aria-label="Mode">
            <button id="btnOrto" class="btn btn-lg btn-light" onclick="setOrtho()"
            title="Ortho/ISO View"><i class="fas fa-cube"></i></button>
            <button id="btnPers" class="btn btn-lg btn-secondary"
            onclick="setPersp()" title="Perspective View"><i class="fab fa-unity"></i>
        </button>
        </div>
        <div class="btn-group" role="group" aria-label="Animation">
            <button id="btnRotate" class="btn btn-lg btn-light"
            onclick="toggleRotate()" title="Toggle Rotate"><i class="fas fa-globe"></i>
        </button>
            <button id="btnQuote" class="btn btn-lg btn-light"
            onclick="toggleQuote()" title="Show Quotation"><i class="fas fa-ruler"></i>
        </button>
            <button id="btnGrid" class="btn btn-lg btn-light"
            onclick="toggleGrid()" title="Toggle Grid"><i class="fas fa-border-all"></i>
        </button>
        </div>
    </div>
    <div id="divMenu2" class="p-2">
        <div>
            <button id="btnHelp" class="btn btn-lg btn-light"
            onclick="toggleHelp()" title="Hide/Show Menu"><i class="fas fa-question"></i>
        </button>
    </div>

```

```

        <div class="btn-group" role="group" aria-label="Animation">
            <button id="btnPause" class="btn btn-lg btn-secondary"
onclick="toggleAnim()" title="Freeze model"><i class="fas fa-pause"></i></button>
            <button id="btnPlay" class="btn btn-lg btn-light"
onclick="toggleAnim()" title="Animation Active"><i class="fas fa-play"></i>
</button>
        </div>
        <button id="btnReset" class="btn btn-lg btn-secondary"
onclick="resetCamera()" title="Reset"><i class="fas fa-sync-alt"></i></button>
    </div>
    <div id = "infoDiv">
        <ul class="list-group list-group-sm small" style="font-size: 0.75em;">
            <li class="list-group-item p-1 active text-start">
                <b>Camera settings</b>
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <b>Zoom</b> Wheel-Scroll
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <b>Pan</b> Drag Mouse-Wheel
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <b>Rotate</b> CTRL + Drag Mouse-Wheel
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <div><b>Iso / Ortho</b></div>
                <div>
                    <i class="fas fa-cube"></i> | key : <b>0</b>
                </div>
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <div><b>Perspective</b></div>
                <div>
                    <i class="fab fa-unity"></i> | key : <b>P</b>
                </div>
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <div><b>AutoRotate</b></div>
                <div>
                    <i class="fas fa-globe"></i> | key : <b>R</b>
                </div>
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <div><b>Quotes</b></div>
                <div>
                    <i class="fas fa-ruler"></i> | key : <b>Q</b>
                </div>
            </li>
            <li class="list-group-item p-1 d-flex justify-content-between">
                <div><b>Grid</b></div>
                <div>
                    <i class="fas fa-border-all"></i> | key : <b>G</b>
                </div>
            </li>
        </ul>
    </div>

```

```

        <li class="list-group-item p-1 d-flex justify-content-between">
            <div><b>Reset</b></div>
            <div>
                <i class="fas fa-sync-alt"></i> | key: <b>SpaceBar</b>
            </div>
        </li>
        <li class="list-group-item p-1 d-flex justify-content-between">
            <div><b>Menu</b></div>
            <div>
                <i class="fas fa-question"></i> | key : <b>H</b>
            </div>
        </li>
    </ul>
</div>
</div>
<div id="divLogo" class="bg-dark bg-opacity-50 bg-gradient p-1 small text-light text-center">
    <img class="img-fluid" src ="Images/LogoEgw.png" height="32" />
    <small>EgalWare</small>
</div>
<div id = "DoorRender"></div>
<div id = "RefRender"></div>
<div id = "Ref1Render"></div>
<div class="fixed-bottom small">
    <div class="d-flex justify-content-between bg-dark text-light">
        <div class="px-2">
            WebDoor 3D
        </div>
        <div class="px-2">
            Egalware | <a class="text-light text-decoration-none"
href="https://www.egalware.com/" target="_blank"></a>
        </div>
    </div>
</div>
</div>

```

@inject IJSRuntime JSRuntime permette di usufruire del servizio che si occupa di eseguire chiamate a funzioni javascript direttamente dal codice c#

## 5 Setup della chiamata alla funzione di render

... probabilmente non necessario con secondo modulo...

Nella pagina Pages/WebGLViewer aggiungere il seguente codice che, una volta effettuato il render della pagina procederà al disegno del visualizzatore.

```

@code {
    protected override async Task OnAfterRenderAsync(bool firstRender)
    {
        await Task.Delay(1);
    }
}

```

```
    if (firstRender)
    {
        var options = new
        {
            dimX = 960,
            dimY = 540,
            fileName = "Cubo.3mf",
            _showCalcGrid = false
        };
        await JSRuntime.InvokeVoidAsync("setup", options);
    }
}
```