Tutorial series: learning how to write a 3D soft engine from scratch in C#, TypeScript or JavaScript
教程系列：学习如何在 C#、TypeScript 或 JavaScript 中从头开始编写 3D 软引擎

June 13, 2013 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/) / David Rousset (https://www.davrous.com/author/davrous/) / 3D Software Engine (https://www.davrous.com/category/3d-software-engine/), English (https://www.davrous.com/category/english/), Technical article (https://www.davrous.com/category/technical-article/), Tutorial (https://www.davrous.com/category/tutorial/)

六月 13， 2013 David Rousset 3D 软件引擎， 英语， 技术文章， 教程

I'd to like to share with you how I've learned to build what's known as a "**3D soft engine**" through a series of tutorials. “Software engine” means that we **will use only the CPU** to build a 3D engine in an old school way (remember Doom on your 80386 ?). I’ll share with you the **C#, TypeScript and JavaScript** versions of the code. In this list, you should then find your favorite language or at least something near your favorite one. The idea is to help you transposing the following samples & concepts on your favorite platform. You’ll find the Visual Studio 2012 C#/TS/JS solutions to download at the end also.

Gouraud Shading on the Suzanne Blender Mesh

我想通过一系列教程与大家分享我是如何学会构建所谓的“3D软引擎”的。“软件引擎”意味着我们将只使用CPU以老派的方式构建3D引擎（还记得80386上的Doom吗？我将与您分享代码的C#，TypeScript和JavaScript版本。在此列表中，您应该找到自己喜欢的语言或至少接近您最喜欢的语言的语言。这个想法是帮助你在你最喜欢的平台上转换以下样本和概念。您还可以在最后找到Visual Studio 2012 C#/TS/JS解决方案的下载。

So why building a 3D soft engine? Well, it’s simply because it really helps understanding how modern 3D works with our GPUs. Indeed, I’m currently learning the basics of 3D thanks to internal workshops delivered within Microsoft by the awesome David Catuhe (https://twitter.com/deltakosh). He’s been mastering 3D for many years now and matrices operations is hard-coded in his brain. When I was young, I was dreaming to be able to write such engines but I had the feeling it was too complex for me. Finally, you’ll see that this is not – that – complex. You simply need someone that will help you understanding the underlying principles in a simple way.

那么为什么要构建3D软引擎呢？嗯，这仅仅是因为它确实有助于理解现代 3D 如何与我们的 GPU 一起工作。事实上，我目前正在学习3D的基础知识，这要归功于令人敬畏的David Catuhe在Microsoft内提供的内部研讨会。他已经掌握3D很多年了，矩阵运算在他的大脑中是硬编码的。当我年轻的时候，我梦想能够编写这样的引擎，但我觉得这对我来说太复杂了。最后，你会发现这并不复杂。您只需要有人以简单的方式帮助您理解基本原则。

Through this series, you will learn how to project some 3D coordinates (X, Y, Z) associated to a point (a vertex) on a 2D screen, how to draw lines between each point, how to fill some triangles, to handle lights, materials and so on. This first tutorial will simply show you how to display 8 points associated to a cube and how to move them in a virtual 3D world.

通过本系列，您将学习如何在 2D 屏幕上投影与点（顶点）关联的一些 3D 坐标（X、Y、Z），如何在每个点之间绘制线条，如何填充一些三角形，处理灯光、材质等。第一个教程将简单地向您展示如何显示与立方体关联的 8 个点以及如何在虚拟 3D 世界中移动它们。

This tutorial is part of the following series:

本教程是以下系列的一部分：

1 – Writing the core logic for camera, mesh & device object (this article)

1 – 为相机、网格和设备对象编写核心逻辑（本文）

2 – Drawing lines and triangles to obtain a wireframe rendering (https://davrous.com/2013/06/14/tutorial-part-2-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-ts-or-js-drawing-lines-triangles/)

2 – 绘制线条和三角形以获得线框渲染

3 – Loading meshes exported from Blender in a JSON format (https://davrous.com/archive/2013/06/17/tutorial-part-3-learning-how-to-write-a-3d-soft-engine-in-c-ts-or-js-loading-meshes-exported-from-blender/)

3 – 加载从 Blender 导出的 JSON 格式的网格

4 – Filling the triangle with rasterization and using a Z-Buffer (https://davrous.com/2013/06/21/tutorial-part-4-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-rasterization-z-buffering/)

4 – 用光栅化填充三角形并使用 Z 缓冲区

4b – Bonus: using tips & parallelism to boost the performance (https://davrous.com/2013/06/25/tutorial-part-4-bonus-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-optimizing-parallelism/)

4b – 奖励：使用提示和并行性来提高性能

5 – Handling light with Flat Shading & Gouraud Shading (https://davrous.com/2013/07/03/tutorial-part-5-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-flat-gouraud-shading/)

5 – 使用平面阴影和 Gouraud 阴影处理光线

6 – Applying textures, back-face culling and WebGL (https://davrous.com/2013/07/18/tutorial-part-6-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-texture-mapping-back-face-culling-webgl/)

6 – 应用纹理、背面剔除和 WebGL

If you're following the complete series, you will **know how to build your own 3D software engine**!

如果您正在学习完整的系列，您将知道如何构建自己的 3D 软件引擎！

Your engine will then start by doing some wireframe rendering, then rasterization followed by gouraud shading and lastly by applying textures like demonstrated in this sample: http://david.blob.core.windows.net/html5/SoftEngineProgression/wireframe/index.html (http://david.blob.core.windows.net/html5/SoftEngineProgression/wireframe/index.html)

然后，您的引擎将首先执行一些线框渲染，然后进行光栅化，然后进行 gouraud 着色，最后应用纹理，如以下示例所示：http://david.blob.core.windows.net/html5/SoftEngineProgression/wireframe/index.html

It's demonstrating the various stage we'll cover during this series going from wireframe to textures.

它演示了我们将在本系列中涵盖从线框到纹理的各个阶段。

By properly following this first tutorial, you'll learn how to rotate the 8 points of a cube to obtain the following result at the end:

通过正确遵循第一个教程，您将学习如何旋转立方体的 8 个点，以在最后获得以下结果：

Disclaimer: some of you are wondering why I'm building this 3D software engine rather than using GPU. It's really for educational purposes. Of course, if you need to build a game with fluid 3D animations, you will need DirectX or OpenGL/WebGL. But once you will have understood how to build a 3D soft engine, more "complex" engine will be simpler to understand. To go further, you definitely should have a look to the BabylonJS WebGL engine (http://www.babylonjs.com/) built by David Catuhe and I. More details & tutorials here: https://doc.babylonjs.com (https://doc.babylonjs.com)

Check the MVA video training version: with David Catuhe, we've made a free 8 modules course to let you learn the basics of 3D, WebGL and Babylon.js (http://www.babylonjs.com/). The first module is containing a 40 min video version of this tutorial series: Introduction to WebGL 3D with HTML5 and Babylon.js (https://doc.babylonjs.com/how_to/3d_on_the_web_understanding_the_basics)

查看MVA视频培训版本：我们与David Catuhe一起制作了一个免费的8个模块课程，让您学习3D，WebGL和Babylon.js的基础知识。第一个模块包含本教程系列的 40 分钟视频版本：使用 HTML5 和 Babylon的 WebGL 3D 简介.js

image (http://www.microsoftvirtualacademy.com/training-courses/introduction-to-webgl-3d-with-html5-and-babylon-js)

Reading prerequisites 阅读先决条件

I've been thinking on how to write these tutorials for a long time now. And I've finally decided not to explain each required principle myself. There is a lot of good resources on the web that will explain those important principles better than I. But I've then spent quite some time browsing the web for you to choose, according to myself, the best one to read:

我一直在思考如何编写这些教程很长时间了。我最终决定不自己解释每个必需的原则。网络上有很多很好的资源可以比我更好地解释这些重要原则。但是我花了相当多的时间浏览网页，供您选择，据我自己说，最好的阅读：

– World, View and Projection Matrix Unveiled (http://web.archive.org/web/20131222170415/http:/robertokoci.com/world-view-projection-matrix-unveiled/)

– 世界、视图和投影矩阵揭幕

– Tutorial 3 : Matrices (http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/) that will provide you an introduction to matrices, the model, view & projection matrices.

教程3：矩阵，将为您提供矩阵，模型，视图和投影矩阵的介绍。

– Cameras on OpenGL ES 2.x – The ModelViewProjection Matrix (http://db-in.com/blog/2011/04/cameras-on-opengl-es-2-x/) : this one is really interesting also as it explains the story starting by how cameras and lenses work.

– OpenGL ES 2.x 上的相机 – 模型视图投影矩阵： 这个也非常有趣，因为它从相机和镜头的工作原理开始解释了故事。

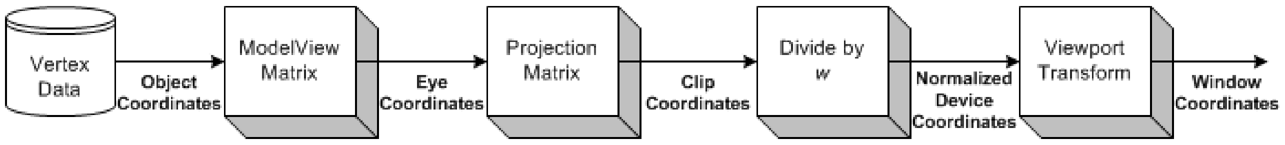
– Transforms (Direct3D 9) (http://msdn.microsoft.com/en-us/library/windows/desktop/bb206269(v=vs.85).aspx)

– 转换 （Direct3D 9）

– A brief introduction to 3D (http://inear.se/talk/a_brief_introduction_to_3d.pptx): an excellent PowerPoint slides deck ! Read at least up to slide 27. After that, it’s too linked to a technology talking to GPU (OpenGL or DirectX).

– 3D简介：一个优秀的PowerPoint幻灯片！至少阅读幻灯片 27。在那之后，它与与GPU（OpenGL或DirectX）通信的技术紧密相连。

– OpenGL Transformation (http://www.songho.ca/opengl/gl_transform.html) – OpenGL 转型



Read those articles by not focusing on the technology associated (like OpenGL or DirectX) or on the concept of triangles you may have seen in the figures. We will see that later on.

阅读这些文章时，不要关注相关的技术（如OpenGL或DirectX）或您可能在图表中看到的三角形概念。我们稍后会看到这一点。

By reading those articles, you really need to understand that there is a series of transformations done that way:

通过阅读这些文章，您确实需要了解以这种方式完成了一系列转换：

– we start by a **3D object centered on itself**

– 我们从一个以自身为中心的 3D 对象开始

– the same object is then **moved into the virtual 3D world** by translation, scaling or rotation operations via matrices

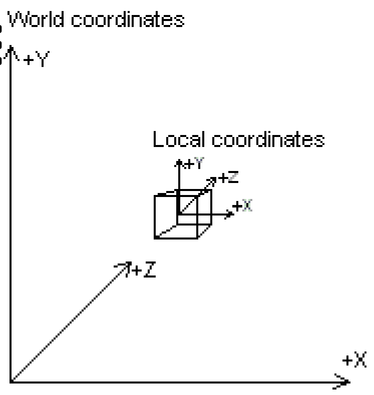
– 然后通过矩阵进行平移、缩放或旋转操作将同一对象移动到虚拟 3D 世界中

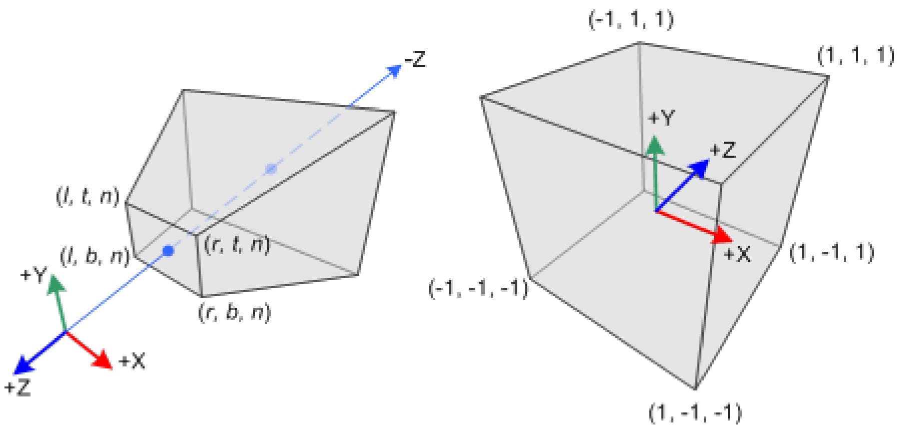
– a **camera will look at this 3D object** positioned in the 3D world

– 相机将查看位于 3D 世界中的这个 3D 对象

– the final **projection** of all that will be done into a **2D space** which is your screen

–将所有操作的最终投影到2D空间中，即您的屏幕





All this magic is done by cumulating transformations through matrices operations. **You should really be at least a bit familiar with those concepts before running through these tutorials**. Even if you don't understand everything by reading them the first time. You should read them first. You will probably go back to those articles later on while writing your own version of this 3D soft engine. This is completely normal, don't worry! 😊 The best way to learn 3D if by experimenting and doing mistakes.

所有这些魔术都是通过矩阵运算累积转换来完成的。在运行这些教程之前，您至少应该熟悉这些概念。即使您第一次阅读它们并不能理解所有内容。您应该先阅读它们。稍后，您可能会在编写自己的3D软引擎版本时返回这些文章。这是完全正常的，别担心！ 😊 通过实验和犯错来学习3D的最佳方式。

We won't neither spend some times on how matrix operations works. The good news is that you don't really need to understand matrices. Simply view it as a black box doing the right operations for you. I'm not a master of matrices but I've managed to write a 3D soft engine by myself. So you should also succeed in doing so.

我们不会花一些时间在矩阵运算的工作原理上。好消息是，你真的不需要理解矩阵。只需将其视为一个黑匣子，为您执行正确的操作。我不是矩阵大师，但我设法自己编写了一个 3D 软引擎。所以你也应该成功地做到这一点。

We will then use libraries that will do the job for us: **SharpDX**, a managed wrapper on top of DirectX, for C# developers and **babylon.math.js** written by David Catuhe for JavaScript developers. I've rewritten it in TypeScript also.

然后，我们将使用可以为我们完成这项工作的库：SharpDX，一个基于DirectX的托管包装器，用于C#开发人员和babylon.math.js由David Catuhe为JavaScript开发人员编写。我也用TypeScript重写了它。

Software prerequisites 软件先决条件

We will write a WinRT/**XAML Windows Store Apps** in C# and/or a **HTML5 application** with TypeScript/JavaScript. So if you want to use the C# samples as-is, you need to install:

我们将用 C# 编写 WinRT/XAML Windows 应用商店应用程序和/或使用 TypeScript/JavaScript 编写 HTML5 应用程序。因此，如果要按原样使用 C# 示例，则需要安装：

1 – Windows 8
2 – Visual Studio 2012 Express for Windows Store Apps. You can download it for free: http://msdn.microsoft.com/en-US/windows/apps/br211386 (http://msdn.microsoft.com/en-US/windows/apps/br211386)

2 – Visual Studio 2012 Express for Windows Store Apps。您可以免费下载：http://msdn.microsoft.com/en-US/windows/apps/br211386

If you choose to use the **TypeScript** samples, you need to install it from: http://www.typescriptlang.org/#Download (http://www.typescriptlang.org/#Download) . All samples have been updated and tested successfully with TypeScript 0.9.

如果选择使用 TypeScript 示例，则需要从以下位置安装它：http://www.typescriptlang.org/#Download 。所有示例均已使用 TypeScript 0.9 成功更新和测试。

You will find the plug-in for Visual Studio 2012 but there are other options available: Sublime Text, Vi, Emacs: TypeScript enabled! (http://msopentech.com/blog/2012/10/01/sublime-text-vi-emacs-typescript-enabled) On my side, I've learned TypeScript by porting the C# version of my code to TypeScript. If you're also interested in learning TypeScript, a first good introduction is this webcast: Anders Hejlsberg: Introducing TypeScript (http://channel9.msdn.com/posts/Anders-Hejlsberg-Introducing-TypeScript) . Please install also Web Essentials 2012 (http://visualstudiogallery.msdn.microsoft.com/07d54d12-7133-4e15-becb-6f451ea3bea6) which had a full support for TypeScript preview and compilation.

你会发现Visual Studio 2012的插件，但还有其他选项可用：Sublime Text，Vi，Emacs：启用TypeScript！就我而言，我通过将代码的C#版本移植到TypeScript来学习TypeScript。如果你也有兴趣学习TypeScript，第一个很好的介绍是这个网络广播：Anders Hejlsberg：介绍TypeScript。请同时安装 Web Essentials 2012，它完全支持 TypeScript 预览和编译。

If you choose **JavaScript**, you just need your favorite IDE and a HTML5 compatible browser. 🙂

如果你选择JavaScript，你只需要你最喜欢的IDE和一个HTML5兼容的浏览器。🙂

Please create a project named "***SoftEngine***" targeting the language you'd like to use. If it's **C#**, add the "**SharpDX core assembly**" by using NuGet on your solution:

请创建一个名为"SoftEngine"的项目，定位您要使用的语言。如果是 C#，请在解决方案上使用 NuGet 添加"SharpDX 核心程序集"：

image (https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/01/10/46/metablogapi/3108.im age_13610CD0.png)

If it's **TypeScript**, download **babylon.math.ts** (http://david.blob.core.windows.net/softengine3d/babylon.math.ts). If' it's **JavaScript** download **babylon.math.js** (http://david.blob.core.windows.net/softengine3d/babylon.math.js). Add a reference to those files in both cases.

如果是 TypeScript，请下载 babylon.math.ts。如果是JavaScript，请下载babylon.math.js。在这两种情况下都添加对这些文件的引用。

Back buffer & rendering loop
后台缓冲区和渲染循环

In a 3D engine, we're rendering the complete scene during each frame with the hope of keeping an optimal 60 frames per second (FPS) to keep fluid animations. To do our rendering job, we need what we call a back buffer. This could be seen as 2 dimensional array mapping the screen/window size. Every cell of the array is mapped to a pixel on the screen.

在3D引擎中，我们在每一帧中渲染完整的场景，希望保持最佳的每秒60帧（FPS）以保持流畅的动画。为了完成渲染工作，我们需要所谓的后台缓冲区。这可以看作是映射屏幕/窗口大小的二维数组。数组的每个单元格都映射到屏幕上的一个像素。

In our **XAML** Windows Store Apps, we will use a byte[] array that will act as our dynamic **back buffer**. For every frame being rendered in the animation loop (tick), this buffer will be affected to a WriteableBitmap acting as the source of a XAML image control that will be called the **front buffer**. For the rendering loop, we're going to ask to the XAML rendering engine to call us for every frame it will generate. The registration is done thanks to this line of code:

在我们的 XAML Windows 应用商店应用中，我们将使用 byte[] 数组作为我们的动态后台缓冲区。对于在动画循环 （tick） 中呈现的每个帧，此缓冲区将受到影响，该位图充当 XAML 图像控件的源，该控件将称为前端缓冲区。对于呈现循环，我们将要求 XAML 呈现引擎为它将生成的每个帧调用我们。注册是通过以下代码行完成的：

```
CompositionTarget.Rendering += CompositionTarget_Rendering;
```

In **HTML5**, we're going to use of course the **<canvas />** element. The canvas element has already a back buffer data array associated to it. You can access it through the ***getImageData()*** and ***setImageData()*** functions. The animation loop will be handled by the *requestAnimationFrame() (http://msdn.microsoft.com/library/ie/hh920765(v=vs.85).aspx)* function. This one is much more efficient that an equivalent of a *setTimeout(function() {},* *1000/60)* as it’s handled natively by the browser that will callback our code only when it will be ready to draw.

在HTML5中，我们当然会使用

__Note:__ in both cases, you can render the frames in a different resolution that the actual width & height of the final window. For instance, you can have a back buffer of 640×480 pixels whereas the final display screen (front buffer) will be in 1920×1080. In XAML and thanks to CSS in HTML5, you will then benefit from “*hardware scaling*”. The rendering engines of XAML and of the browser will stretch the back buffer data to the front buffer window by even using an anti-aliasing algorithm. In both cases, this task is done by the GPU. This is why we call it “hardware scaling” (hardware is the GPU). You can read more about this topic addressed in HTML5 here: Unleash the power of HTML 5 Canvas for gaming (http://blogs.msdn.com/b/eternalcoding/archive/2012/03/22/unleash-the-power-of-html-5-canvas-for-gaming-part-1.aspx?Redirected=true) . This approach is often used in games for instance to boost the performance as you have less pixels to address.

注意：在这两种情况下，您都可以以与最终窗口的实际宽度和高度不同的分辨率渲染帧。例如，您可以有一个 640×480 像素的后台缓冲区，而最终的显示屏（前缓冲区）将是 1920×1080。在XAML中，由于HTML5中的CSS，您将受益于“硬件扩展”。XAML 和浏览器的呈现引擎甚至会使用抗锯齿算法将后台缓冲区数据拉伸到前端缓冲区窗口。在这两种情况下，此任务都由 GPU 完成。这就是为什么我们称之为“硬件扩展”（硬件是GPU）。您可以在此处阅读有关 HTML5 中解决的此主题的更多信息：释放 HTML 5 Canvas 在游戏中的强大功能。这种方法通常用于游戏，例如，由于要解决的像素较少，因此可以提高性能。

Camera & Mesh objects
相机和网格对象

Let’s start coding. First, we need to define some objects that will embed the details needed for a camera and for a mesh. A mesh is a cool name to describe a 3D object.

让我们开始编码。首先，我们需要定义一些对象，这些对象将嵌入摄像机和网格体所需的细节。网格是描述 3D 对象的一个很酷的名称。

Our *Camera* will have 2 properties: its position in the 3D world and where it’s looking at, the target. Both are made of 3D coordinates named a Vector3. C# will use **SharpDX.Vector3** and TypeScript & JavaScript will use **BABYLON.Vector3**.

我们的相机将具有2个属性：它在3D世界中的位置以及它正在看的位置，目标。两者都由名为 Vector3 的 3D 坐标组成。C#将使用SharpDX.Vector3，TypeScript和JavaScript将使用 BABYLON。矢量3.

Our *Mesh* will have a collection of vertices (several vertex or 3D points) that will be used to [redacted]d our 3D object, its position in the 3D world and its rotation state. To identify it, it will also have a name.

我们的网格将具有一组顶点（几个顶点或3D点），这些顶点将用于构建我们的3D对象，其在3D世界中的位置及其旋转状态。为了识别它，它还将有一个名称。

To resume, we need the following code:

要恢复，我们需要以下代码：

- C#
- TypeScript
- JavaScript JavaScript

```
// Camera.cs & Mesh.cs
using SharpDX;

namespace SoftEngine
{
    public class Camera
    {
        public Vector3 Position { get; set; }
        public Vector3 Target { get; set; }
    }
    public class Mesh
    {
        public string Name { get; set; }
        public Vector3[] Vertices { get; private set; }
        public Vector3 Position { get; set; }
        public Vector3 Rotation { get; set; }

        public Mesh(string name, int verticesCount)
        {
            Vertices = new Vector3[verticesCount];
            Name = name;
        }
    }
}
```

```
//<reference path="babylon.math.ts"/>
module SoftEngine {
    export class Camera {
        Position: BABYLON.Vector3;
        Target: BABYLON.Vector3;

        constructor() {
            this.Position = BABYLON.Vector3.Zero();
            this.Target = BABYLON.Vector3.Zero();
        }
    }
    export class Mesh {
        Position: BABYLON.Vector3;
        Rotation: BABYLON.Vector3;
        Vertices: BABYLON.Vector3[];

        constructor(public name: string, verticesCount: number) {
            this.Vertices = new Array(verticesCount);
            this.Rotation = BABYLON.Vector3.Zero();
            this.Position = BABYLON.Vector3.Zero();
        }
    }
}
```

```
var SoftEngine;
(function (SoftEngine) {
    var Camera = (function () {
        function Camera() {
            this.Position = BABYLON.Vector3.Zero();
            this.Target = BABYLON.Vector3.Zero();
        }
        return Camera;
    })();
    SoftEngine.Camera = Camera;
    var Mesh = (function () {
        function Mesh(name, verticesCount) {
            this.name = name;
            this.Vertices = new Array(verticesCount);
            this.Rotation = BABYLON.Vector3.Zero();
            this.Position = BABYLON.Vector3.Zero();
        }
        return Mesh;
    })();
    SoftEngine.Mesh = Mesh;
})(SoftEngine || (SoftEngine = {}));
```

For instance, if you want to describe a cube using our Mesh object, you need to create 8 vertices associated to the 8 points of the cube. Here are the coordinates on a cube displayed in Blender:

例如，如果您想使用我们的 Mesh 对象描述立方体，则需要创建与立方体的 8 个点关联的 8 个顶点。以下是在Blender中显示的立方体上的坐标：

image (https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/01/10/46/metablogapi/4540.image_2D374352.png)

With a left-handed world. Remember also that when you're creating a mesh, the coordinates system is starting at the center of the mesh. So, X=0, Y=0, Z=0 is the center of the cube.

有一个左撇子的世界。还要记住，创建网格时，坐标系从网格的中心开始。所以，X=0，Y=0，Z=0是立方体的中心。

This could be created via this kind of code:

这可以通过这种代码创建：

```
var mesh = new Mesh("Cube", 8);
mesh.Vertices[0] = new Vector3(-1, 1, 1);
mesh.Vertices[1] = new Vector3(1, 1, 1);
mesh.Vertices[2] = new Vector3(-1, -1, 1);
mesh.Vertices[3] = new Vector3(-1, -1, -1);
mesh.Vertices[4] = new Vector3(-1, 1, -1);
mesh.Vertices[5] = new Vector3(1, 1, -1);
mesh.Vertices[6] = new Vector3(1, -1, 1);
mesh.Vertices[7] = new Vector3(1, -1, -1);
```

The most important part: the Device object
最重要的部分：设备对象

Now that we have our basic objects and we know how to build 3D meshes, we need the most important part: the **Device object. It's the core of our 3D engine**.

现在我们有了基本对象并且知道如何构建 3D 网格，我们需要最重要的部分：Device 对象。这是我们3D引擎的核心。

In it's rendering function, we will build the view matrix and the projection matrix based on the camera we will have defined before.

在它的渲染函数中，我们将基于我们之前定义的相机构建视图矩阵和投影矩阵。

Then, we will iterate through each available mesh to build their associated world matrix based on their current rotation and translation values. Finally, once done, the final transformation matrix to apply is:

然后，我们将遍历每个可用的网格，以根据其当前的旋转和平移值构建其关联的世界矩阵。最后，一旦完成，要应用的最终转换矩阵是：

```
var transformMatrix = worldMatrix * viewMatrix * projectionMatrix;
```

This is the concept you absolutely need to understand by reading the previous prerequisites resources. Otherwise, you will probably simply copy/paste the code without understanding anything about the magic underneath. This is not a very big problem for further tutorials but again, it's better to know what's you're coding.

这是您绝对需要通过阅读前面的先决条件资源来理解的概念。否则，您可能只是复制/粘贴代码，而不了解下面的魔术。对于进一步的教程来说，这不是一个很大的问题，但同样，最好知道你在编码什么。

Using this transformation matrix, we're going to project each vertex of each mesh in the 2D world to obtain X,Y coordinates from their X,Y,Z coordinates. To finally draw on screen, we're adding a small clip logic to only display visible pixels via a PutPixel method/function.

使用此转换矩阵，我们将投影 2D 世界中每个网格的每个顶点，以从其 X，Y，Z 坐标获取 X，Y 坐标。为了最终在屏幕上绘制，我们添加了一个小的剪辑逻辑，以仅通过 PutPixel 方法/函数显示可见像素。

Here are the various versions of the Device object. I've tried to comment the code to help you understanding it as much as possible.

下面是设备对象的各种版本。我尝试注释代码以帮助您尽可能多地理解它。

<u>**Note:**</u> Microsoft Windows is drawing using the BGRA color space (Blue, Green, Red, Alpha) whereas the HTML5 canvas is drawing using the RGBA (http://en.wikipedia.org/wiki/RGBA_color_space) (Red, Green, Blue, Alpha) color space. That's why, you will notice some slight differences in the code between C# and HTML5.

注意：Microsoft Windows 使用 BGRA 色彩空间（蓝色、绿色、红色、Alpha）进行绘制，而 HTML5 画布使用 RGBA（红色、绿色、蓝色、Alpha）色彩空间进行绘制。这就是为什么你会注意到C#和HTML5之间的代码有一些细微的差异。

- C#
- TypeScript
- JavaScript JavaScript

```
using Windows.UI.Xaml.Media.Imaging;
using System.Runtime.InteropServices.WindowsRuntime;
using SharpDX;

namespace SoftEngine
{
    public class Device
    {
        private byte[] backBuffer;
        private WriteableBitmap bmp;

        public Device(WriteableBitmap bmp)
        {
            this.bmp = bmp;
            // the back buffer size is equal to the number of pixels to draw
            // on screen (width*height) * 4 (R,G,B & Alpha values).
            backBuffer = new byte[bmp.PixelWidth * bmp.PixelHeight * 4];
        }

        // This method is called to clear the back buffer with a specific color
        public void Clear(byte r, byte g, byte b, byte a) {
            for (var index = 0; index < backBuffer.Length; index += 4)
            {
                // BGRA is used by Windows instead by RGBA in HTML5
                backBuffer[index] = b;
                backBuffer[index + 1] = g;
                backBuffer[index + 2] = r;
                backBuffer[index + 3] = a;
            }
        }

        // Once everything is ready, we can flush the back buffer
        // into the front buffer.
        public void Present()
        {
            using (var stream = bmp.PixelBuffer.AsStream())
            {
                // writing our byte[] back buffer into our WriteableBitmap stream
                stream.Write(backBuffer, 0, backBuffer.Length);
            }
            // request a redraw of the entire bitmap
            bmp.Invalidate();
        }

        // Called to put a pixel on screen at a specific X,Y coordinates
        public void PutPixel(int x, int y, Color4 color)
        {
            // As we have a 1-D Array for our back buffer
            // we need to know the equivalent cell in 1-D based
            // on the 2D coordinates on screen
            var index = (x + y * bmp.PixelWidth) * 4;

            backBuffer[index] = (byte)(color.Blue * 255);
            backBuffer[index + 1] = (byte)(color.Green * 255);
            backBuffer[index + 2] = (byte)(color.Red * 255);
            backBuffer[index + 3] = (byte)(color.Alpha * 255);
        }

        // Project takes some 3D coordinates and transform them
        // in 2D coordinates using the transformation matrix
        public Vector2 Project(Vector3 coord, Matrix transMat)
        {
            // transforming the coordinates
            var point = Vector3.TransformCoordinate(coord, transMat);
            // The transformed coordinates will be based on coordinate system
            // starting on the center of the screen. But drawing on screen normally start
            // from top left. We then need to transform them again to have x:0, y:0 on to
            var x = point.X * bmp.PixelWidth + bmp.PixelWidth / 2.0f;
            var y = -point.Y * bmp.PixelHeight + bmp.PixelHeight / 2.0f;
            return (new Vector2(x, y));
        }

        // DrawPoint calls PutPixel but does the clipping operation before
        public void DrawPoint(Vector2 point)
        {
            // Clipping what's visible on screen
            if (point.X >= 0 && point.Y >= 0 && point.X < bmp.PixelWidth && point.Y < bmp
            {
                // Drawing a yellow point
                PutPixel((int)point.X, (int)point.Y, new Color4(1.0f, 1.0f, 0.0f, 1.0f))
            }
        }

        // The main method of the engine that re-compute each vertex projection
        // during each frame
        public void Render(Camera camera, params Mesh[] meshes)
        {
            // To understand this part, please read the prerequisites resources
            var viewMatrix = Matrix.LookAtLH(camera.Position, camera.Target, Vector3.Uni
            var projectionMatrix = Matrix.PerspectiveFovRH(0.78f,
                                                           (float)bmp.PixelWidth / bmp.P
                                                           0.01f, 1.0f);

            foreach (Mesh mesh in meshes)
            {
                // Beware to apply rotation before translation
                var worldMatrix = Matrix.RotationYawPitchRoll(mesh.Rotation.Y,
                                                              mesh.Rotation.X, mesh.Rotat
                                  Matrix.Translation(mesh.Position);

                var transformMatrix = worldMatrix * viewMatrix * projectionMatrix;

                foreach (var vertex in mesh.Vertices)
                {
                    // First, we project the 3D coordinates into the 2D space
                    var point = Project(vertex, transformMatrix);
                    // Then we can draw on screen
                    DrawPoint(point);
                }
            }
        }
    }
}
```

```
///<reference path="babylon.math.ts"/>

module SoftEngine {
    export class Device {
        // the back buffer size is equal to the number of pixels to draw
        // on screen (width*height) * 4 (R,G,B & Alpha values).
        private backbuffer: ImageData;
        private workingCanvas: HTMLCanvasElement;
        private workingContext: CanvasRenderingContext2D;
        private workingWidth: number;
        private workingHeight: number;
        // equals to backbuffer.data
        private backbufferdata;

        constructor(canvas: HTMLCanvasElement) {
            this.workingCanvas = canvas;
            this.workingWidth = canvas.width;
            this.workingHeight = canvas.height;
            this.workingContext = this.workingCanvas.getContext("2d");
        }

        // This function is called to clear the back buffer with a specific color
        public clear(): void {
            // Clearing with black color by default
            this.workingContext.clearRect(0, 0, this.workingWidth, this.workingHeight);
            // once cleared with black pixels, we're getting back the associated image da
            // clear out back buffer
            this.backbuffer = this.workingContext.getImageData(0, 0, this.workingWidth, t
        }

        // Once everything is ready, we can flush the back buffer
        // into the front buffer.
        public present(): void {
            this.workingContext.putImageData(this.backbuffer, 0, 0);
        }

        // Called to put a pixel on screen at a specific X,Y coordinates
        public putPixel(x: number, y: number, color: BABYLON.Color4): void {
            this.backbufferdata = this.backbuffer.data;
            // As we have a 1-D Array for our back buffer
            // we need to know the equivalent cell index in 1-D based
            // on the 2D coordinates of the screen
            var index: number = ((x >> 0) + (y >> 0) * this.workingWidth) * 4;



            // RGBA color space is used by the HTML5 canvas
            this.backbufferdata[index] = color.r * 255;
            this.backbufferdata[index + 1] = color.g * 255;
            this.backbufferdata[index + 2] = color.b * 255;
            this.backbufferdata[index + 3] = color.a * 255;
        }

        // Project takes some 3D coordinates and transform them
        // in 2D coordinates using the transformation matrix
        public project(coord: BABYLON.Vector3, transMat: BABYLON.Matrix): BABYLON.Vector2
            // transforming the coordinates
            var point = BABYLON.Vector3.TransformCoordinates(coord, transMat);
            // The transformed coordinates will be based on coordinate system
            // starting on the center of the screen. But drawing on screen normally start
            // from top left. We then need to transform them again to have x:0, y:0 on to
            var x = point.x * this.workingWidth + this.workingWidth / 2.0 >> 0;
            var y = -point.y * this.workingHeight + this.workingHeight / 2.0 >> 0;
            return (new BABYLON.Vector2(x, y));
        }

        // drawPoint calls putPixel but does the clipping operation before
        public drawPoint(point: BABYLON.Vector2): void {
            // Clipping what's visible on screen
            if (point.x >= 0 && point.y >= 0 && point.x < this.workingWidth
                                            && point.y < this.workingHeight) {
                // Drawing a yellow point
                this.putPixel(point.x, point.y, new BABYLON.Color4(1, 1, 0, 1));
            }
        }

        // The main method of the engine that re-compute each vertex projection
        // during each frame
        public render(camera: Camera, meshes: Mesh[]): void {
            // To understand this part, please read the prerequisites resources
            var viewMatrix = BABYLON.Matrix.LookAtLH(camera.Position, camera.Target, BAB
            var projectionMatrix = BABYLON.Matrix.PerspectiveFovLH(0.78,
                                                 this.workingWidth / this.workingHeight

            for (var index = 0; index < meshes.length; index++) {
                // current mesh to work on
                var cMesh = meshes[index];
                // Beware to apply rotation before translation
                var worldMatrix = BABYLON.Matrix.RotationYawPitchRoll(
                        cMesh.Rotation.y, cMesh.Rotation.x, cMesh.Rotation.z)
                        .multiply(BABYLON.Matrix.Translation(
                            cMesh.Position.x, cMesh.Position.y, cMesh.Position.z));

                var transformMatrix = worldMatrix.multiply(viewMatrix).multiply(projectio

                for (var indexVertices = 0; indexVertices < cMesh.Vertices.length; index
                    // First, we project the 3D coordinates into the 2D space
                    var projectedPoint = this.project(cMesh.Vertices[indexVertices], tra
                    // Then we can draw on screen
                    this.drawPoint(projectedPoint);
                }
            }
        }
    }
}
```

```
var SoftEngine;
(function (SoftEngine) {
    var Device = (function () {
        function Device(canvas) {
            // Note: the back buffer size is equal to the number of pixels to draw
            // on screen (width*height) * 4 (R,G,B & Alpha values).
            this.workingCanvas = canvas;
            this.workingWidth = canvas.width;
            this.workingHeight = canvas.height;
            this.workingContext = this.workingCanvas.getContext("2d");
        }

        // This function is called to clear the back buffer with a specific color
        Device.prototype.clear = function () {
            // Clearing with black color by default
            this.workingContext.clearRect(0, 0, this.workingWidth, this.workingHeight);
            // once cleared with black pixels, we're getting back the associated image da
            // clear out back buffer
            this.backbuffer = this.workingContext.getImageData(0, 0, this.workingWidth, t
        };

        // Once everything is ready, we can flush the back buffer
        // into the front buffer.
        Device.prototype.present = function () {
            this.workingContext.putImageData(this.backbuffer, 0, 0);
        };

        // Called to put a pixel on screen at a specific X,Y coordinates
        Device.prototype.putPixel = function (x, y, color) {
            this.backbufferdata = this.backbuffer.data;
            // As we have a 1-D Array for our back buffer
            // we need to know the equivalent cell index in 1-D based
            // on the 2D coordinates of the screen
            var index = ((x >> 0) + (y >> 0) * this.workingWidth) * 4;



            // RGBA color space is used by the HTML5 canvas
            this.backbufferdata[index] = color.r * 255;
            this.backbufferdata[index + 1] = color.g * 255;
            this.backbufferdata[index + 2] = color.b * 255;
            this.backbufferdata[index + 3] = color.a * 255;
        };

        // Project takes some 3D coordinates and transform them
        // in 2D coordinates using the transformation matrix
        Device.prototype.project = function (coord, transMat) {
            var point = BABYLON.Vector3.TransformCoordinates(coord, transMat);
            // The transformed coordinates will be based on coordinate system
            // starting on the center of the screen. But drawing on screen normally start
            // from top left. We then need to transform them again to have x:0, y:0 on to
            var x = point.x * this.workingWidth + this.workingWidth / 2.0 >> 0;
            var y = -point.y * this.workingHeight + this.workingHeight / 2.0 >> 0;
            return (new BABYLON.Vector2(x, y));
        };

        // drawPoint calls putPixel but does the clipping operation before
        Device.prototype.drawPoint = function (point) {
            // Clipping what's visible on screen
            if (point.x >= 0 && point.y >= 0 && point.x < this.workingWidth
                                                && point.y < this.workingHeight) {
                // Drawing a yellow point
                this.putPixel(point.x, point.y, new BABYLON.Color4(1, 1, 0, 1));
            }
        };

        // The main method of the engine that re-compute each vertex projection
        // during each frame
        Device.prototype.render = function (camera, meshes) {
            // To understand this part, please read the prerequisites resources
            var viewMatrix = BABYLON.Matrix.LookAtLH(camera.Position, camera.Target, BABY
            var projectionMatrix = BABYLON.Matrix.PerspectiveFovLH(0.78,
                                              this.workingWidth / this.workingHeight, 0.01,

            for (var index = 0; index < meshes.length; index++) {
                // current mesh to work on
                var cMesh = meshes[index];
                // Beware to apply rotation before translation
                var worldMatrix = BABYLON.Matrix.RotationYawPitchRoll(
                    cMesh.Rotation.y, cMesh.Rotation.x, cMesh.Rotation.z)
                     .multiply(BABYLON.Matrix.Translation(
                       cMesh.Position.x, cMesh.Position.y, cMesh.Position.z));

                var transformMatrix = worldMatrix.multiply(viewMatrix).multiply(projectic

                for (var indexVertices = 0; indexVertices < cMesh.Vertices.length; index\
                    // First, we project the 3D coordinates into the 2D space
                    var projectedPoint = this.project(cMesh.Vertices[indexVertices], trar
                    // Then we can draw on screen
                    this.drawPoint(projectedPoint);
                }
            }
        };
        return Device;
    })();
    SoftEngine.Device = Device;
})(SoftEngine || (SoftEngine = {}));
```

Putting it all together
将一切整合在一起

We finally need to create a mesh (our cube), create a camera and target our mesh & instantiate our Device object.

我们最终需要创建一个网格（我们的立方体），创建一个摄像机并定位我们的网格并实例化我们的设备对象。

Once done, we will launch the animation/rendering loop. In optimal cases, this loop will be called every 16ms (60 FPS). During each tick (call to the handler registered to the rendering loop), we will launch the following logic every time:

完成后，我们将启动动画/渲染循环。在最佳情况下，此循环将每 16 毫秒 （60 FPS） 调用一次。在每个刻度（调用注册到渲染循环的处理程序）期间，我们每次都会启动以下逻辑：

1 – **Clear the screen** and all associated pixels with black ones (*Clear()* function)

1 – 清除屏幕和所有带有黑色像素的关联像素（Clear（） 函数）

2 – **Update the various position & rotation values** of our meshes

2 – 更新网格的各种位置和旋转值

3 – **Render them** into the back buffer by doing the required matrix operations (*Render()* function)

3 – 通过执行所需的矩阵运算（Render（） 函数）将它们渲染到后台缓冲区中

4 – **Display them** on screen by flushing the back buffer data into the front buffer (*Present()* function)

4 – 通过将后台缓冲区数据刷新到前台缓冲区（Present（） 函数）在屏幕上显示它们

- C#
- TypeScript
- JavaScript JavaScript

```
private Device device;
Mesh mesh = new Mesh("Cube", 8);
Camera mera = new Camera();

private void Page_Loaded(object sender, RoutedEventArgs e)
{
    // Choose the back buffer resolution here
    WriteableBitmap bmp = new WriteableBitmap(640, 480);

    device = new Device(bmp);

    // Our XAML Image control
    frontBuffer.Source = bmp;

    mesh.Vertices[0] = new Vector3(-1, 1, 1);
    mesh.Vertices[1] = new Vector3(1, 1, 1);
    mesh.Vertices[2] = new Vector3(-1, -1, 1);
    mesh.Vertices[3] = new Vector3(-1, -1, -1);
    mesh.Vertices[4] = new Vector3(-1, 1, -1);
    mesh.Vertices[5] = new Vector3(1, 1, -1);
    mesh.Vertices[6] = new Vector3(1, -1, 1);
    mesh.Vertices[7] = new Vector3(1, -1, -1);

    mera.Position = new Vector3(0, 0, 10.0f);
    mera.Target = Vector3.Zero;

    // Registering to the XAML rendering loop
    CompositionTarget.Rendering += CompositionTarget_Rendering;
}

// Rendering loop handler
void CompositionTarget_Rendering(object sender, object e)
{
    device.Clear(0, 0, 0, 255);

    // rotating slightly the cube during each frame rendered
    mesh.Rotation = new Vector3(mesh.Rotation.X + 0.01f, mesh.Rotation.Y + 0.01f, mesh.Ro

    // Doing the various matrix operations
    device.Render(mera, mesh);
    // Flushing the back buffer into the front buffer
    device.Present();
}
```

```
///<reference path="SoftEngine.ts"/>

var canvas: HTMLCanvasElement;
var device: SoftEngine.Device;
var mesh: SoftEngine.Mesh;
var meshes: SoftEngine.Mesh[] = [];
var mera: SoftEngine.Camera;

document.addEventListener("DOMContentLoaded", init, false);

function init() {
    canvas = <HTMLCanvasElement> document.getElementById("frontBuffer");
    mesh = new SoftEngine.Mesh("Cube", 8);
    meshes.push(mesh);
    mera = new SoftEngine.Camera();
    device = new SoftEngine.Device(canvas);

    mesh.Vertices[0] = new BABYLON.Vector3(-1, 1, 1);
    mesh.Vertices[1] = new BABYLON.Vector3(1, 1, 1);
    mesh.Vertices[2] = new BABYLON.Vector3(-1, -1, 1);
    mesh.Vertices[3] = new BABYLON.Vector3(-1, -1, -1);
    mesh.Vertices[4] = new BABYLON.Vector3(-1, 1, -1);
    mesh.Vertices[5] = new BABYLON.Vector3(1, 1, -1);
    mesh.Vertices[6] = new BABYLON.Vector3(1, -1, 1);
    mesh.Vertices[7] = new BABYLON.Vector3(1, -1, -1);

    mera.Position = new BABYLON.Vector3(0, 0, 10);
    mera.Target = new BABYLON.Vector3(0, 0, 0);

    // Calling the HTML5 rendering loop
    requestAnimationFrame(drawingLoop);
}

// Rendering loop handler
function drawingLoop() {
    device.clear();

    // rotating slightly the cube during each frame rendered
    mesh.Rotation.x += 0.01;
    mesh.Rotation.y += 0.01;

    // Doing the various matrix operations
    device.render(mera, meshes);
    // Flushing the back buffer into the front buffer
    device.present();

    // Calling the HTML5 rendering loop recursively
    requestAnimationFrame(drawingLoop);
}
```

```
var canvas;
var device;
var mesh;
var meshes = [];
var mera;

document.addEventListener("DOMContentLoaded", init, false);

function init() {
    canvas = document.getElementById("frontBuffer");
    mesh = new SoftEngine.Mesh("Cube", 8);
    meshes.push(mesh);
    mera = new SoftEngine.Camera();
    device = new SoftEngine.Device(canvas);

    mesh.Vertices[0] = new BABYLON.Vector3(-1, 1, 1);
    mesh.Vertices[1] = new BABYLON.Vector3(1, 1, 1);
    mesh.Vertices[2] = new BABYLON.Vector3(-1, -1, 1);
    mesh.Vertices[3] = new BABYLON.Vector3(-1, -1, -1);
    mesh.Vertices[4] = new BABYLON.Vector3(-1, 1, -1);
    mesh.Vertices[5] = new BABYLON.Vector3(1, 1, -1);
    mesh.Vertices[6] = new BABYLON.Vector3(1, -1, 1);
    mesh.Vertices[7] = new BABYLON.Vector3(1, -1, -1);

    mera.Position = new BABYLON.Vector3(0, 0, 10);
    mera.Target = new BABYLON.Vector3(0, 0, 0);

    // Calling the HTML5 rendering loop
    requestAnimationFrame(drawingLoop);
}

// Rendering loop handler
function drawingLoop() {
    device.clear();

    // rotating slightly the cube during each frame rendered
    mesh.Rotation.x += 0.01;
    mesh.Rotation.y += 0.01;

    // Doing the various matrix operations
    device.render(mera, meshes);
    // Flushing the back buffer into the front buffer
    device.present();

    // Calling the HTML5 rendering loop recursively
    requestAnimationFrame(drawingLoop);
}
```

If you’ve managed to follow properly this first tutorial, you should obtain something like that:

如果您设法正确遵循了第一个教程，则应获得类似以下内容：

If not, **download the solutions** containing the source code:

如果没有，请下载包含源代码的解决方案：

– **C#** : SoftEngineCSharpPart1.zip (http://david.blob.core.windows.net/softengine3d/SoftEngineCSharpPart1.zip)

C＃： SoftEngineCSharpPart1.zip

– **TypeScript** : SoftEngineTSPart1.zip (http://david.blob.core.windows.net/softengine3d/SoftEngineTSPart1.zip)

– 打字稿： 软引擎TSPart1.zip

– **JavaScript** : SoftEngineJSPart1.zip (http://david.blob.core.windows.net/softengine3d/SoftEngineJSPart1.zip) or simply right-click –> view source on the embedded iframe

JavaScript：SoftEngineJSPart1.zip或者只需右键单击嵌入式iframe上的>查看源代码

Simply review the code and try to find what's wrong with yours. 🙄

只需查看代码并尝试找出您的问题。 🙄

In the next tutorial, we're going to **learn how to draw lines between each vertex** & the concept of faces/**triangles** to obtain something like that:

在下一个教程中，我们将学习如何在每个顶点之间绘制线条以及面/三角形的概念以获得类似的东西：

image (https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/01/10/46/metablogapi/8228.image_445B9E1D.png)

See you in the second part of this series: Tutorial part 2: learning how to write a 3D soft engine from scratch in C#, TS or JS – drawing lines & triangles (https://davrous.com/2013/06/14/tutorial-part-2-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-ts-or-js-drawing-lines-triangles/)

本系列的第二部分见：教程第2部分：学习如何用C#，TS或JS从头开始编写3D软引擎 - 绘制线条和三角形

Follow the author @davrous

~~关注~~作者@davrous (https://twitter.com/davrous)

Tagged 3DEngine (https://www.davrous.com/tag/3dengine/), C# (https://www.davrous.com/tag/c/), Canvas (https://www.davrous.com/tag/canvas/), GPU (https://www.davrous.com/tag/gpu/), HTML5 (https://www.davrous.com/tag/html5/), JavaScript (https://www.davrous.com/tag/javascript/), Tutorial (https://www.davrous.com/tag/tutorial/), TypeScript (https://www.davrous.com/tag/typescript/), Windows 8 (https://www.davrous.com/tag/windows-8/)

HTML5 Gaming: benchmarking your sprites animations to target all devices & browsers

HTML5 Gaming：对精灵动画进行基准测试，以针对所有设备和浏览器

(https://www.davrous.com/2013/04/26/html5-gaming-benchmarking-your-sprites-animations-to-target-all-devices-browsers/)

Tutorial part 2: learning how to write a 3D soft engine from scratch in C#, TS or JS – drawing lines & triangles

教程第2部分：学习如何用C#，TS或JS从头开始编写3D软引擎 - 绘制线条和三角形

(https://www.davrous.com/2013/06/14/tutorial-part-2-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-ts-or-js-drawing-lines-triangles/)

76 thoughts on “Tutorial series: learning how to write a 3D soft engine from scratch in C#, TypeScript or JavaScript”

关于“教程系列：学习如何用 C#、TypeScript 或 JavaScript 从头开始编写 3D 软引擎”的 76 条思考

demonixis says: 恶魔 说：

June 13, 2013 at 1:50 pm

六月 13， 2013 1：50 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-658)

Hi ! 你好!

Thanks for this great introduction on how to make a soft 3D engine ! It's really easy to read, I think it's good to offer these three languages because it allows us to see several methods. Another big thanks, I can't wait for the next =)

感谢您对如何制作软 3D 引擎的精彩介绍！它真的很容易阅读，我认为提供这三种语言很好，因为它让我们可以看到几种方法。另一个非常感谢，我等不及下一个=)

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=658#respond)

Orhan says: 奥尔罕 说：
June 18, 2013 at 11:47 pm
六月 18， 2013 11：47 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-659)

Hello, 你好

The JS demo doesn't work with firefox, error "requestAnimationFrame is not defined"

JS演示不适用于火狐浏览器，错误"requestAnimationFrame未定义"

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=659#respond)

Alejandro Godofredo Carlstein Ramos Mejia (http://acarlstein.com) says:
罗·戈多弗雷多·卡尔斯坦·拉莫斯·梅希亚说：
January 18, 2018 at 8:45 pm
一月 18， 2018 在 8：45 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-16173)

Try the following: 请尝试以下操作：

TypeScript:

window.requestAnimationFrame = (function(){

window.requestAnimationFrame = （function（）{

return window.requestAnimationFrame ||

返回窗口.请求动画帧 ||

(window).webkitRequestAnimationFrame ||

（窗口）.webkit请求动画框架 ||

(window).mozRequestAnimationFrame ||

（窗口）.mozRequestAnimationFrame ||

(window).oRequestAnimationFrame ||

（窗口）.o请求动画帧 ||

(window).msRequestAnimationFrame ||

（窗口）.ms请求动画帧 | |

function(/* function */ callback, /* DOMElement */ element){

function（/* function */ callback， /* DOMElement */ element）{

window.setTimeout(callback, 1000 / 60);

window.setTimeout（回调， 1000 / 60）；

};
})();

JavaScript: JavaScript：

window.requestAnimationFrame = (function () {

window.requestAnimationFrame = （function （） {

return window.requestAnimationFrame | |

返回窗口.请求动画帧 | |

window.webkitRequestAnimationFrame | |

window.webkitRequestAnimationFrame | |

window.mozRequestAnimationFrame | |

window.mozRequestAnimationFrame | |

window.oRequestAnimationFrame | |

window.o请求动画帧 | |

window.msRequestAnimationFrame | |

窗口.ms请求动画帧 | |

function (/* function */ callback, /* DOMElement */ element) {

function （/* function */ callback， /* DOMElement */ element） {

window.setTimeout(callback, 1000 / 60);

window.setTimeout（回调， 1000 / 60）；

};
})();

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=16173#respond)

Alejandro Godofredo Carlstein Ramos Mejia (http://acarlstein.com) says:

罗·戈多弗雷多·卡尔斯坦·拉莫斯·梅希亚说：

January 18, 2018 at 8:48 pm

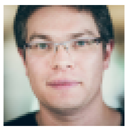
一月 18， 2018 在 8：48 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-16174)

For the TypeScript code didn't show up properly on the comments. So, I will try again:

因为 TypeScript 代码没有正确显示在注释中。所以，我会再试一次：

```
window.requestAnimationFrame = (function(){
window.requestAnimationFrame = （function（）{
return window.requestAnimationFrame ||
返回窗口.请求动画帧 ||
(<any> window).webkitRequestAnimationFrame ||
（窗口）.webkit请求动画框架 ||
(<any> window).mozRequestAnimationFrame ||
（窗口）.mozRequestAnimationFrame ||
(<any> window).oRequestAnimationFrame ||
（窗口）.o请求动画帧 ||
(<any> window).msRequestAnimationFrame ||
（窗口）.ms请求动画帧 ||
function(/* function */ callback, /* DOMElement */ element){
function（/* function */ callback， /* DOMElement */ element）{
window.setTimeout(callback, 1000 / 60);
window.setTimeout（回调， 1000 / 60）;
};
})();
```

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=16174#respond)



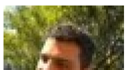
davrous says: 达夫鲁斯 说：
June 19, 2013 at 12:13 am
六月 19， 2013 12：13 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-660)

Hello Orhan. Thanks for the feedback. I'll fix that asap.

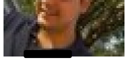
你好奥尔罕。感谢您的反馈。我会尽快解决这个问题。

David 大卫

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=660#respond)



Lucas says: 卢卡斯 说：


January 25, 2019 at 2:44 pm

一月 25， 2019 在 2：44 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-20984)

This is THE best article I've read!

这是我读过的最好的文章！

Ps: Please update the links, some of them are not working!

PS：请更新链接，其中一些不起作用！

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=20984#respond)

Zuff says: 扎夫 说：

June 19, 2013 at 2:43 am

六月 19， 2013 2：43 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-661)

for JS : 对于 JS：

It should be added at the beginning of this script main.js file (for three examples):

它应该添加到此脚本 main.js 文件的开头（对于三个示例）：

// shim layer with setTimeout fallback

具有设置超时回退的填充层

window.requestAnimationFrame = (function () {

window.requestAnimationFrame = （function （） {

  return window.requestAnimationFrame ||

返回窗口.请求动画帧 ||

      window.webkitRequestAnimationFrame ||

window.webkitRequestAnimationFrame ||

      window.mozRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

      function (callback) {

函数（回调）{

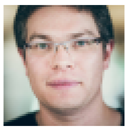
        window.setTimeout(callback, 1000 / 60);

window.setTimeout（回调， 1000 / 60）；

      };

  })();

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=661#respond)


davrous says: 达夫鲁斯 说：
June 19, 2013 at 2:48 am

六月 19， 2013 2：48 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-662)

Hi guys, well, the shim for RAF was already in place. So which version of Firefox are you using?

嗨，伙计们，好吧，英国皇家空军的垫片已经到位。那么你使用的是哪个版本的火狐？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=662#respond)

Zuff says: 扎夫 说：
June 19, 2013 at 4:39 am

六月 19， 2013 4：39 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-663)

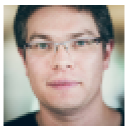
Le problème ne vient pas de ton site "david.blob.core.windows.net/softengine3d" (http://david.blob.core.windows.net/softengine3d") qui fonctionne (testé sur Maxthon 3, IE9, chrome 27, FirexFox 15.0.1, 16.0.2 et 21.0) mais de tes zip pour javascript.

问题不是来自你的"david.blob.core.windows.net/softengine3d"网站（在Maxthon 3，IE9，chrome 27，FirexFox 15.0.1，16.0.2和21.0上测试），而是来自javascript的zip。

Pour preuve, il parle d'un problème avec la fonction "requestAnimationFrame" (requestAnimationFrame(drawingLoop);) alors que celle que tu utilises sur ton site s'appelle "requestAnimFrame" (c'est grâce à IE que j'ai pu récupérer justement le script manquant dans le main.js en faisant une comparaison de code).

作为证据，他谈到了"requestAnimationFrame"函数（requestAnimationFrame（drawingLoop））的问题;)而您在网站上使用的那个称为"requestAnimFrame"（多亏了 IE，我才能通过代码比较精确地恢复手中丢失的脚本.js）。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=663#respond)

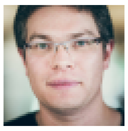

davrous says: 达夫鲁斯 说：
June 19, 2013 at 4:51 am

六月 19， 2013 4：51 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-664)

Ah ok. Je comprends mieux. 🙂 Bon, je modifie les archive ZIP alors. Merci pour l'info.

啊，~~好吧~~。我更明白了。🙂 好的，我正在编辑ZIP档案。感谢您提供的信息。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=664#respond)



davrous says: 达夫鲁斯 说:
June 19, 2013 at 5:01 am

六月 19， 2013 5：01 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-665)

I've updated the .ZIP files containing the JavaScript code to include requestAnimationFrame prefixed for Firefox & iDevices.

我已经更新了包含JavaScript代码的.ZIP文件，以包含为Firefox和iDevices前缀的requestAnimationFrame。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=665#respond)

Lucas says: 卢卡斯 说:
July 3, 2013 at 5:36 am

七月 3， 2013 5：36 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-666)

Really good work David. I really like your approach – writing engine from scratch even with front end part where You put pixels on screen.

真的很好的工作大卫。我真的很喜欢你的方法——从头开始编写引擎，即使是你在屏幕上放置像素的前端部分。

I'm wondering – if it's possible to write it in C# in VS2010 under win 7? Without using XAML and whole "web environment" ?

我想知道 - 是否可以在win 2010下的win 7下用C#编写它？不使用 XAML 和整个“网络环境”？

SharpDX is useful for vectors and matrix operations – but how do I "link" it with my project under VS 2010? Also what should I use rather than XAML image control?

SharpDX 对于矢量和矩阵运算很有用 – 但是如何在 VS 2010 下将其与我的项目“链接”？另外，我应该使用什么而不是 XAML 图像控件?

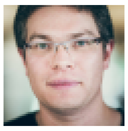
I would be really grateful for some advices 😉

我将不胜感激一些建议 😉

Take care 当心

Lucas 卢卡斯

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=666#respond)



davrous says: 达夫鲁斯 说:
July 3, 2013 at 6:02 am

七月 3， 2013 6：02 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-667)

Hi Lucas. Thanks for your feedback! Yes, you can. I know some people who has already port the tutorials in C#/Winform, in Java and even Under Windows CE.

嗨，卢卡斯。感谢您的反馈！是的，你可以。我知道有些人已经在C#/Winform，Java甚至Windows CE下移植了教程。

You can download the SharpDX version working with Win7 desktop project: http://sharpdx.org/ (http://sharpdx.org/)

您可以下载使用 Win7 桌面项目的 SharpDX 版本：http://sharpdx.org/

David 大卫

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=667#respond)

Hastur says: 哈斯图尔 说:
July 11, 2013 at 6:06 am

七月 11， 2013 6：06 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-668)

Super ! 超!

J'ai enfin fini les prérequis et donc cette première partie. J'ai adoré. Par contre, je suis frustré car je n'ai toujours pas compris, au sens mathématique, pourquoi pour construire la matrice de projection il faut diviser les composantes x et y des points par leur composante z. Mais bon, on peut vivre sans avoir compris ça et au pire je le reprendrai plus tard.

我终于完成了先决条件，所以这是第一部分。我喜欢它。另一方面，我很沮丧，因为我仍然不明白，在数学意义上，为什么要构建投影矩阵，你必须将点的 x 和 y 分量除以它们的 z 分量。但是，嘿，我们可以不理解这一点而生活，最坏的情况是我稍后会收回它。

Encore merci et vivement la suite 🙂

再次感谢您，期待其余的 🙂

@Lucas : I'm also on a Win7 environement, and I made a WPF project so David's code is 99% compatible. The only difference is that WriteableBimap has no "PixelBuffer" property. The workaround is the WritePixels() method.

@Lucas：我也在Win7环境中，我做了一个WPF项目，所以David的代码是99%兼容的。唯一的区别是WriteableBimap没有“PixelBuffer”属性。解决方法是 WritePixels（） 方法。

So my Present() method contains only the following code :

所以我的 Present（） 方法只包含以下代码：

_bitmap.WritePixels(new System.Windows.Int32Rect(0, 0, _bitmap.PixelWidth, _bitmap.PixelHeight), _backBuffer, _bitmap.PixelWidth * 4, 0);

_位图。WritePixels（new System.Windows.Int32Rect（0， 0， _bitmap.像素宽度，_bitmap。像素高度）、_backBuffer、_bitmap。像素宽度 * 4， 0）；

The other difference is in the init of the WritableBitmap :

另一个区别在于 WritableBitmap 的 init：

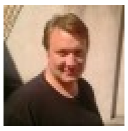
var bmp = new WriteableBitmap(640, 480, 96, 96, PixelFormats.Bgra32, null);

var bmp = new WriteableBitmap（640， 480， 96， 96， PixelFormats.Bgra32， null）；

96 is the "dpi" for x and y, I've found the value somewhere on the web and I guess it's the default dpi for a screen but I don't know if it's supposed to vary depending on the screen. Anyway the code works great !

96 是 x 和 y 的“dpi”，我在网络上的某个地方找到了这个值，我想它是屏幕的默认 dpi，但我不知道它是否应该根据屏幕而变化。无论如何，代码效果很好！

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=668#respond)



Richard says: 理查德 说：
December 13, 2015 at 6:16 pm

十二月 13， 2015 6：16 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-696)

Hi Lucas, 嗨卢卡斯，

How did you get around the lack of Invalidate() on WriteableBitMap in wpf?

你是如何解决 wpf 中 WriteableBitMap 上缺少 Invalidate（） 的问题的？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=696#respond)

Jared says: 贾里德 说：
March 19, 2017 at 6:14 pm

三月19，2017 6时：下午14 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-8733)

For anyone else that is currently wondering about this. This is how I did it in WPF:

对于目前对此感到疑惑的其他任何人。这是我在 WPF 中的做法：

public void Present(Image image)

公共空隙 现在（图片）

{
bmp.WritePixels(new System.Windows.Int32Rect(0, 0, bmp.PixelWidth, bmp.PixelHeight), backBuffer, bmp.PixelWidth * 4, 0);

.bmp。WritePixels（new System.Windows.Int32Rect（0， 0， bmp.像素宽度，bmp。PixelHeight），backBuffer，bmp。像素宽度 * 4， 0）；

image.Source = bmp; 图像。源 = bmp;
}
I have an Image on the main page that I pass to this method. bmp is still a WriteableBitmap. My image is 1920×1080 and it writes in less than a millisecond with my setup.

我在主页上有一个传递给此方法的图像。bmp 仍然是一个可写位图。我的图像是 1920×1080，使用我的设置在不到一毫秒的时间内写入。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=8733#respond)

Lucas says: 卢卡斯 说：
July 12, 2013 at 3:35 am

七月 12， 2013 3：35 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-669)

@Hastur

thank you for information 🙂

谢谢你的信息 🙂

Can You tell me something about performance of yours solution? I'm not sure but I've heard that WritePiexels() method is not very fast 😉

您能告诉我有关您的解决方案性能的一些信息吗？我不确定，但我听说 WritePiexels（） 方法不是很快 😉

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=669#respond)

Hastur says: 哈斯图尔 说：
August 19, 2013 at 1:17 am

八月 19， 2013 1：17 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-670)

Oops, sorry Lucas but I didn't see you answered something until now. So if you come by here again, here is my answer : I was at 60fps with the cube on an Intel G870 CPU (my workstation), and still almost at 60fps with the textured monkey of the last tuto.

哎呀，对不起卢卡斯，但直到现在我才看到你回答了什么。因此，如果您再次来到这里，这是我的答案：我在英特尔 G60 CPU（我的工作站）上使用立方体的速度为 870fps，而最后一个 tuto 的纹理猴子仍然几乎为 60fps。

I tried to access directly the WriteableBitmap's backbuffer like David, using unsafe code like in this link : stackoverflow.com/.../how-to-edit-a-writablebitmap-backbuffer-in-non-ui-thread (http://stackoverflow.com/questions/9868929/how-to-edit-a-writablebitmap-backbuffer-in-non-ui-thread)

我尝试像大卫一样直接访问 WriteableBitmap 的后台缓冲区，使用不安全的代码，如以下链接：stackoverflow.com/.../how-to-edit-a-writablebitmap-backbuffer-in-non-ui-thread

I was like "Wow ! I'm using unsafe code ! I'm a reel demomaker, it's gonna be über fast !!". XD

我就像"哇！我正在使用不安全的代码！我是一个卷轴演示者，它会很快！！鑫达

And, big surprise, it was awfully slow.

而且，令人惊讶的是，它非常慢。

So to my experience, WritePixels() is way more efficient than direct memory access to the backbuffer, but maybe it exists another method I haven't heard about.

因此，根据我的经验，WritePixels（） 比直接访问后台缓冲区的内存更有效，但也许它存在另一种我没有听说过的方法。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=670#respond)

Saad Galib says: 你可以得到 加利布 说：

September 22, 2013 at 3:30 am

九月 22， 2013 3：30 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-671)

My problem is in the device.prototype.project() in the javascript solution in the following lines,

我的问题在javascript解决方案中的device.prototype.project（）中，在以下几行中，

```
    var x = point.x * this.workingWidth + this.workingWidth / 2.0 >> 0;
```

var x = point.x * this.workingWidth + this.workingWidth / 2.0 >> 0;

```
    var y = -point.y * this.workingHeight + this.workingHeight / 2.0 >> 0;
```

var y = -point.y * this.workingHeight + this.workingHeight / 2.0 >> 0;

I don't understand the multiplication with workingHeight and width. This may be pretty trivial, but i am stuck 😟 any help would be much appreciated 🙂

我不明白工作高度和宽度的乘法。这可能很简单，但我被困住了😟，任何帮助将不胜感激 🙂

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=671#respond)

---

cneveu@teaser.fr says: cneveu@teaser.fr 说：

October 16, 2013 at 1:27 am

十月 16， 2013 1：27 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-672)

Great, simply great !

太好了，简直太好了！

I've been trying to add another camera, without success, if anyone wants to give it a try.

我一直在尝试添加另一台相机，但没有成功，如果有人想尝试一下。

I'm actively working on those tutorials, and would like to extend them into another blog/discussion for a deeper approach/explanation if I'm allowed to.

我正在积极研究这些教程，如果允许的话，我想将它们扩展到另一个博客/讨论中，以获得更深入的方法/解释。

Anyway very good job. (See you at the tech days.)

总之，干得非常好。（科技日见。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=672#respond)

Hastur says: 哈斯图尔 说：

October 17, 2013 at 1:51 am

十月 17， 2013 1：51 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-673)

@Saad : take a look at the two first graphics of the "Reading Prerequisites" chapter.

@Saad：请看"阅读必备"一章的前两个图。

After all transformations and cliping, and just before the transformation to "window space", all coordinates are "normalized", which means they are between -1 and 1.

在所有转换和裁剪之后，就在转换为"窗口空间"之前，所有坐标都被"规范化"，这意味着它们介于 -1 和 1 之间。

So the code you pasted makes sure the final coordinates are between 0 and respectively layer's width and height.

因此，您粘贴的代码可确保最终坐标分别介于 0 和图层的宽度和高度之间。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=673#respond)

Yannick says: 雅尼克 说：
October 21, 2013 at 5:17 am
十月 21， 2013 5：17 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-674)

Bonjour Mr. Rousset, 卓乔尔·鲁塞特先生，

Petite question, est-il possible de développer une application 3D autre qu'un jeu avec ce moteur. J'ai eu utiliser SilverlightXNA qui permettait de faire des interfaces avec tout les contrôles habituels et de plsu y ajouter la 3D.

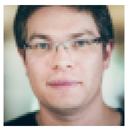
小问题，是否可以使用此引擎开发游戏以外的 3D 应用程序。我不得不使用SilverlightXNA，它允许与所有常用控件制作界面并添加3D。

Qu'en est-il aujourd'hui ?

今天是什么情况?

Merci 谢谢

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=674#respond)



davrous says: 达夫鲁斯 说：
October 21, 2013 at 5:24 am
十月 21， 2013 5：24 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-675)

Bonjour Yannick, 你好雅尼克，

Bah comme tout moteur 3d, il n'y a rien de spécifique au jeu dans ce moteur. Vous pouvez tout à fait l'utiliser sur autre chose que du jeu pour modéliser des graphiques ou toutes autres choses "plus sérieuses" 🙂

Bah 像任何 3D 引擎一样，这个引擎中没有任何特定于游戏的内容。您可以在游戏以外的其他设备上使用它来建模图形或任何其他“更严重” 🙂 的东西

Qu'imaginez-vous réaliser?

你想象实现什么?

Bye, 再见

David 大卫

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=675#respond)

---

cneveu@teaser.fr says: cneveu@teaser.fr 说:
October 24, 2013 at 8:49 am
十月 24， 2013 8：49 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-676)

Bonjour.

J'ai réussi (c'était une erreur de positionnement en Xaml, ca n'est pas ma tasse de Csharp 😊 à ajouter une deuxième caméra qui montre les modèle simultanément sous un autre angle, donc… C'est super sympa à voir)

我成功了（这是 Xaml 中的定位错误，添加第二个从另一个角度同时显示模型的相机不是我的一杯 Csharp 😊，所以……超级好看）

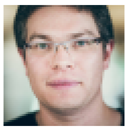
Je suis sous VS2010. Mon appli est en Wpf, (la même quasi).

我在VS2010下。我的应用程序在 WPF 中（几乎相同）。

Si quelqu'un souhaite l'avoir je l'enverrai avec plaisir : mistralkriss at jaimeil (gmail) point com)

如果有人希望拥有它，我将很高兴地发送它：mistralkriss at jaimeil （gmail） dot com)

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=676#respond)



davrous says: 达夫鲁斯 说:
October 24, 2013 at 8:54 am
十月 24， 2013 8：54 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-677)

Cool! 😊 N'hésites pas à mettre ta solution en téléchargement quelque part.

凉！ 😊 请随时在某处下载您的解决方案。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=677#respond)

---

rick says: 里克 说:
October 31, 2013 at 1:06 pm
十月 31， 2013 1：06 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-678)

Hi! Thanks for a great tutorial. Made a lot of sense!

你好！感谢您提供精彩的教程。很有道理！

When I implemented the code everything works perfectly except that the x-axis is mixed up. If I plot two pixels A=[0,0,0] and B[1,0,0] I would expect B to be to the right of A, but it draws to the left of it. Is this expected or have I missed something?

当我实现代码时，除了 x 轴混淆之外，一切都运行良好。如果我绘制两个像素 A=[0，0，0] 和 B[1，0，0]，我希望 B 在 A 的右侧，但它绘制在它的左侧。这是意料之中的还是我错过了什么？

I implemented this with Javascript

我用Javascript实现了这一点

Thanks! 谢谢！

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=678#respond)

James says: 詹姆斯 说：
November 3, 2013 at 11:54 am

十一月 3， 2013 11：54 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-679)

Hi David, 嗨大卫，

Firstly, many thanks for this excellent series of posts! Unfortunately, I've fallen at the first hurdle and hope you might be able to help me up :). I had assumed that the call to TransformCoordinate() in Project() would return normalised X/Y co-ordinates in the range [-1,1], however the range is actually [-0.5, 0.5]! I've looked at the SharpDX code that builds the projection matrix and to my untrained eyes it looks like it should generate X & Y coordinates in the range [-1,1] after the perspective divide – but it doesn't! Do you know why this isn't the case? Also what's the range of the z-value?

首先，非常感谢这一系列优秀的帖子！ 不幸的是，我在第一个障碍上摔倒了，希望你能:)帮助我。 我假设在 Project（） 中调用 TransformCoordinate（） 会返回 [-1，1] 范围内的规范化 X/Y 坐标，但范围实际上是 [-0.5， 0.5]！ 我看过构建投影矩阵的 SharpDX 代码，在我未经训练的眼睛看来，它应该在透视分度后生成 [-1，1] 范围内的 X 和 Y 坐标——但事实并非如此！ 你知道为什么不是这样吗？ 还有 z 值的范围是多少？

Thank you! 谢谢！

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=679#respond)

James says: 詹姆斯 说：
November 4, 2013 at 2:24 am

十一月 4， 2013 2：24 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-680)

This is just to clarify my eariler post. There are two things I'm not quite getting about the mapping to screen space:

这只是为了澄清我的耳帖。 关于映射到屏幕空间，我不太了解两件事：

1) I thought that the clip/perspective matrix generated by the call below would map camera space z-values to -1 at the near clip plane and 1 at the far clip plane; however I've found that changing the far clip plane parameter (e.g. to 100) has no effect on the transformed point. Can you tell me why this is?

1） 我认为下面调用生成的剪辑/透视矩阵会将相机空间 z 值映射到近剪辑平面的 -1 和远剪辑平面的 1;但是我发现更改远裁剪平面参数（例如更改为 100）对转换后的点没有影响。 你能告诉我这是为什么吗？

var projectionMatrix = Matrix.PerspectiveFovRH(0.78f,

var projectionMatrix = Matrix.PerspectiveFovRH（0.78f，

(float)bmp.PixelWidth / bmp.PixelHeight,

（浮点）BMP。像素宽度/bmp。像素高度，

0.01f, 1.0f); 0.01f， 1.0f）；

2) I was also under the impression that the same matrix would map x and y screen space values to the range [-1, 1] (-1 at left and bottom clip planes, 1 at top and bottom), but unless I'm mistaken the code below assumes values in the range [-0.5, 0.5]:

2）我还认为相同的矩阵会将x和y屏幕空间值映射到范围[-1，1]（左侧和底部剪辑平面为-1，顶部和底部为1），但除非我弄错了，否则下面的代码假定值在[-0.5，0.5]范围内：

var x = point.X * bmp.PixelWidth + bmp.PixelWidth / 2.0f;

变量 x = 点。X*bmp。像素宽度 + bmp。像素宽度 / 2.0f;

var y = -point.Y * bmp.PixelHeight + bmp.PixelHeight / 2.0f;

var y = -period。Y * bmp.像素高度 + bmp。像素高度 / 2.0f;

I really want to understand this stuff in order to move onto the next tutorial in the series, therefore please can you let me know why the expectations above proved wrong?

我真的很想了解这些东西，以便进入该系列的下一个教程，因此请您告诉我为什么上述期望被证明是错误的吗？

Many thanks, 有很多感谢，

James 詹姆斯

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=680#respond)

---

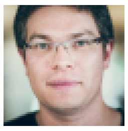
max million says: 最大百万 说：

December 13, 2013 at 7:31 am

十二月 13， 2013 7：31 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-681) babylon.math.js. Add a reference to those files in both cases. (how do you add this file into visual studio library with out breaking the file structure) add it?

babylon.math.js. 在这两种情况下都添加对这些文件的引用。（如何将此文件添加到Visual Studio Library中而不会破坏文件结构）添加它?

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=681#respond)



davrous says: 达夫鲁斯 说:
December 13, 2013 at 7:38 am
十二月 13， 2013 7：38 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-682)

Hi Max, 嗨麦克斯,

I didn't understand your question. Can you please clarify what your problem is?

我不明白你的问题。你能澄清一下你的问题是什么吗?

Thanks, 谢谢

David 大卫

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=682#respond)

gladskih@live.ru says: gladskih@live.ru 说:
January 19, 2014 at 3:48 am
在19 2014：3上午一月48 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-683)

It's the best computer graphic (polygonal modeling) guide I've ever found and it have examples exactly in the languages that I using to learn it. But there are some points disturbing me. First of all it is using additional libraries like Babylon. During learning I would like to reinvent all possible wheels including matrix operations and JSON parsing. Also WinRT is good, but while I know only WinForms I prefer going through the WPF for beginning. And why do you use such insane nested way to create object in JS? Has it any advantages before defining constructor-like function and using "new" statement?

这是我找到的最好的计算机图形（多边形建模）指南，它有我用来学习它的语言的示例。但有几点让我感到不安。首先，它使用像巴比伦这样的其他库。在学习过程中，我想重新发明所有可能的轮子，包括矩阵运算和 JSON 解析。WinRT也很好，但是虽然我只知道WinForms，但我更喜欢通过WPF开始。为什么使用这种疯狂的嵌套方式在JS中创建对象？在定义类似构造函数和使用“new”语句之前有什么优势吗?

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=683#respond)

Ggg says: 格格 说：
March 18, 2014 at 12:05 am
三月 18， 2014 12：05 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-684)

Nice one 好东西

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=684#respond)

Vincent Ellis says: 文森特·埃利斯 说：
April 14, 2014 at 9:06 am
四月 14， 2014 9：06 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-685)

Hello and thanks for the amazing series. It is very well written and everything is well explained. Do you have any plans of adding C++ examples?

您好，感谢您的精彩系列。它写得很好，一切都解释得很好。您有添加C++示例的计划吗？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=685#respond)

sieuro8mig says: Sieuro8mig 说：
May 1, 2014 at 10:21 am
五月 1， 2014 10：21 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-686)

Everything works fine until I let the camera fly through a meshe's origin. Then everything is displayed mirrored but it shouldn't be visible. Is there a way to fix that?

一切正常，直到我让相机飞过一个meshe的起源。然后，所有内容都会镜像显示，但不应可见。有没有办法解决这个问题？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=686#respond)

Shawn says: 肖恩 说：

June 22, 2014 at 4:31 am

六月 22， 2014 4：31 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-687)

Given the formulat: 给定公式：

var x = point.X * bmp.PixelWidth + bmp.PixelWidth / 2.0f;

变量 x = 点。X*bmp。像素宽度 + bmp。像素宽度 / 2.0f;

var y = -point.Y * bmp.PixelHeight + bmp.PixelHeight / 2.0f;

var y = -period。Y * bmp.像素高度 + bmp。像素高度 / 2.0f;

The point [1, 0] will be mapped to [1.5*bmp.PixelWidth, bmp.PixelHeight / 2.0f], so the point will not be draw?

点 [1， 0] 将映射到 [1.5*bmp。像素宽度，bmp。像素高度/ 2.0f]，所以点不会被画吗?

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=687#respond)

CarlV says: 卡尔V说：
July 13, 2014 at 9:37 am

七月 13， 2014 9：37 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-688)

Hi.

Thanks David for the tut ! It really makes things simple to understand.

谢谢大卫的嘟嘟！它确实使事情变得简单易懂。

For this particular one, I've used notepad++ and Javascript.

对于这个特定的，我使用了notepad++和Javascript。

the 8 points animate well in google, firefox; However in ie11, i get the canvas without animation.

这 8 分在谷歌、火狐中表现得很好;但是在 ie11 中，我得到的画布没有动画。

I'm still checking out what went wrong…Any suggestion ?

我还在检查出了什么问题…有什么建议吗?

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=688#respond)

Khalid says: 哈立德 说：
August 14, 2014 at 2:30 am

八月 14， 2014 2：30 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-689)

Hi,

Thank you for the tutorial. Is there a problem with the first pre-requistes link "World, View and Projection Matrix Unveiled"? does anyone have the correct link or the tutorial itself?

感谢您的教程。第一个先决条件链接“世界、视图和投影矩阵揭幕”有问题吗？有没有人有正确的链接或教程本身？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=689#respond)

Vegeta897 says: 贝吉塔897 说：
January 1, 2015 at 5:56 pm
一月 1， 2015 5：56 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-690)

Khalid, you can view the article using the internet archive.

哈立德，您可以使用互联网档案查看文章。

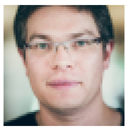
web.archive.org/…/world-view-projection-matrix-unveiled

web.archive.org/…/world-view-projection-matrix-unveiled (http://web.archive.org/web/20131222170415/http://robertokoci.com/world-view-projection-matrix-unveiled/)

This link should be edited into the article, if the author is reading this comment.

如果作者正在阅读此评论，则应将此链接编辑到文章中。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=690#respond)



davrous says: 达夫鲁斯 说：
January 6, 2015 at 5:01 am
在 6， 2015 5：01 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-691)

Thanks for sharing the archive article. I've updated the article.

感谢您分享存档文章。我已经更新了文章。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=691#respond)

Yayasakhar Nahasamapetilan says:
尔·纳哈萨马佩蒂兰 说：

January 28, 2015 at 3:41 pm
一月 28， 2015 3：41 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-692)

Very nice, although I would really like it if this was a bit more language agnostic. I'd rather build everything from the ground up than use libraries.

非常好，虽然如果这更与语言无关，我真的很喜欢它。我宁愿从头开始构建所有内容，也不愿使用库。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=692#respond)

Rudi says: 鲁迪 说：
March 7, 2015 at 6:44 am
三月 7， 2015 6：44 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-693)

Hi.

I created a Windows Store App, writing in C# in Visual Studio Pro 2013 Update4 on Windows8.1. Pro.

我创建了一个Windows 应用商店应用程序，在 Windows8.1 上的 Visual Studio Pro 2013 Update4 中用 C# 编写。亲。

Starting the run, I receive error messages: "frontBuffer" (in the Page_Loaded section) is an unknown name to the project SoftEngine.Windows as is the component "Sharp.DX.D3DCompiler" which is referred to. I think, all references shown in the tutorial are added in my solution.

开始运行时，我收到错误消息："frontBuffer"（在Page_Loaded部分中）是项目SoftEngine.Windows的未知名称，所引用的组件"Sharp.DX.D3DCompiler"也是如此。我认为，教程中显示的所有引用都添加到我的解决方案中。

In the object catalogue I could not find the references for eliminating the error messages.

在对象目录中，我找不到用于消除错误消息的参考。

Can anyone help to fix this?

任何人都可以帮助解决这个问题吗?

Thanks in advance. 提前谢谢。

Rudi 鲁迪

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=693#respond)

Brendan Burkhart says: 布伦丹·伯克哈特 说：
March 5, 2017 at 4:27 pm

三月5，2017 4时：下午27 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-8347)

I’m having this issue too. Did you find a fix for it?

我也遇到了这个问题。你找到修复了吗？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=8347#respond)

Raisun says: 雷舜 说：
October 18, 2017 at 3:04 pm

十月18，2017 3时：下午04 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-14574)

I had the same issue. You should define in your MainPage.xaml file

我有同样的问题。您应该在 MainPage.xaml 文件中定义

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=14574#respond)

Marco says: 马可 说：
September 15, 2015 at 1:09 am

九月 15， 2015 1：09 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-694)

This tutorial is great! Big thanks!

本教程很棒！非常感谢！

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=694#respond)

---

jjcat says: JJCAT 说：
November 16, 2015 at 9:42 am

十一月 16， 2015 9：42 上午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-695) var viewMatrix = Matrix.LookAtLH(camera.Position, camera.Target, Vector3.UnitY);

var viewMatrix = Matrix.LookAtLH（camera.位置，相机。目标，矢量3.UnitY）；

var projectionMatrix = Matrix.PerspectiveFovRH(0.78f, (float)bmp.PixelWidth / bmp.PixelHeight, 0.01f, 1.0f);

var projectionMatrix = Matrix.PerspectiveFovRH（0.78f，（float）bmp.像素宽度/bmp。像素高度，0.01f， 1.0f）；

The first one is using left-hand coordinate system method and the second one is using right-hand, is it mistake?

第一个是使用左手坐标系法，第二个是使用右手坐标系法，是不是错了？

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=695#respond)

---

duke (http://photoatomic.com) says: 公爵说：
March 6, 2016 at 6:03 pm
三月 6， 2016 6：03 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-697)

remembers me 1998! 🙂 the starting point for everyone interested in 3d graphic 🙂

记得我1998！ 🙂 每个对3D图形🙂感兴趣的人的起点

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=697#respond)

Pingback: Tutorial part 6: learning how to write a 3D software engine in C#, TS or JS – Texture mapping, back-face culling & WebGL | David Rousset – HTML5 & Gaming Technical Evangelist (https://blogs.msdn.microsoft.com/davrous/2013/07/18/tutorial-part-6-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-texture-mapping-back-face-culling-webgl/)

Pingback：教程第6部分：学习如何用C#，TS或JS编写3D软件引擎 - 纹理映射，背面剔除和WebGL |David Rousset – HTML5 & Gaming Technical Evangelist

---

Pingback: Tutorial part 2: learning how to write a 3D soft engine from scratch in C#, TS or JS – drawing lines & triangles | David Rousset – HTML5 & Gaming Technical Evangelist (https://blogs.msdn.microsoft.com/davrous/2013/06/14/tutorial-part-2-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-ts-or-js-drawing-lines-triangles/)

Pingback：教程第 2 部分：学习如何在 C#、TS 或 JS 中从头开始编写 3D 软引擎 – 绘制线条和三角形 |David Rousset – HTML5 & Gaming Technical Evangelist

Pingback: Tutorial part 3: learning how to write a 3D soft engine in C#, TS or JS – loading meshes exported from Blender | David Rousset – HTML5 & Gaming Technical Evangelist (https://blogs.msdn.microsoft.com/davrous/2013/06/17/tutorial-part-3-learning-how-to-write-a-3d-soft-engine-in-c-ts-or-js-loading-meshes-exported-from-blender/)

Pingback：教程第 3 部分：学习如何在 C#、TS 或 JS 中编写 3D 软引擎 – 加载从 Blender 导出的网格 |David Rousset – HTML5 & Gaming Technical Evangelist

Pingback: Tutorial part 4: learning how to write a 3D software engine in C#, TS or JS – Rasterization & Z-Buffering | David Rousset – HTML5 & Gaming Technical Evangelist (https://blogs.msdn.microsoft.com/davrous/2013/06/21/tutorial-part-4-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-rasterization-z-buffering/)

Pingback： 教程第 4 部分：学习如何用 C#、TS 或 JS 编写 3D 软件引擎 – 光栅化和 Z 缓冲 |David Rousset – HTML5 & Gaming Technical Evangelist

Pingback: Tutorial part 5: learning how to write a 3D software engine in C#, TS or JS – Flat & Gouraud Shading | David Rousset – HTML5 & Gaming Technical Evangelist (https://blogs.msdn.microsoft.com/davrous/2013/07/03/tutorial-part-5-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-flat-gouraud-shading/)

Pingback： 教程第 5 部分：学习如何用 C#、TS 或 JS 编写 3D 软件引擎 – Flat & Gouraud Shading |David Rousset – HTML5 & Gaming Technical Evangelist

Pingback: Tutorial part 2: learning how to write a 3D soft engine from scratch in C#, TS or JS – drawing lines & triangles – David Rousset (https://www.davrous.com/2013/06/14/tutorial-part-2-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-ts-or-js-drawing-lines-triangles/)

Pingback：教程第2部分：学习如何用C#，TS或JS从头开始编写3D软引擎 - 绘制线条和三角形 - David Rousset

Pingback: Tutorial part 5: learning how to write a 3D software engine in C#, TS or JS – Flat & Gouraud Shading – David Rousset (https://www.davrous.com/2013/07/03/tutorial-part-5-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-flat-gouraud-shading/)

Pingback：教程第 5 部分：学习如何用 C#、TS 或 JS 编写 3D 软件引擎 – Flat & Gouraud Shading – David Rousset

Pingback: Tutorial part 6: learning how to write a 3D software engine in C#, TS or JS – Texture mapping, back-face culling & WebGL – David Rousset (https://www.davrous.com/2013/07/18/tutorial-part-6-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-texture-mapping-back-face-culling-webgl/)

Pingback：教程第6部分：学习如何用C#，TS或JS编写3D软件引擎 - 纹理映射，背面剔除和WebGL - David Rousset

Pingback: Tutorial part 4: learning how to write a 3D software engine in C#, TS or JS – Rasterization & Z-Buffering – David Rousset (https://www.davrous.com/2013/06/21/tutorial-part-4-learning-how-to-write-a-3d-software-engine-in-c-ts-or-js-rasterization-z-buffering/)

Pingback：教程第4部分：学习如何用C#，TS或JS编写3D软件引擎 - 光栅化和Z-Buffering - David Rousset

Joe Valdivia says: 乔·瓦尔迪维亚 说：
May 29, 2016 at 1:10 am
五月29，2016 1在：上午10 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-1292)

public void Present() 公共无效 礼物（）
I am trying to run this code in C# WPF I am stuck here'. It doesn't seem to like "PixelBuffer" or "Invalidate" I know it has to do with WritableBitmap in WPF

我正在尝试在 C# WPF 中运行此代码，我被困在这里。它似乎不喜欢"PixelBuffer"或"Invalidate"，我知道它与WPF中的WritableBitmap有关

Any ideas ? By the way awesome tutorial.

有什么想法吗？顺便说一下，很棒的教程。

{
using (var stream = bmp.PixelBuffer.AsStream())

使用 （var 流 = BMP。PixelBuffer.AsStream（））

{
// writing our byte[] back buffer into our WriteableBitmap stream

将我们的 byte[] 后台缓冲区写入我们的可写位图流

stream.Write(backBuffer, 0, backBuffer.Length);

流。Write（backBuffer， 0， backBuffer.Length）；

}
// request a redraw of the entire bitmap

请求重绘整个位图

bmp.Invalidate(); .bmp。无效（）;
}

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=1292#respond)

---

joe valdivia says: 乔·瓦尔迪维亚 说：
May 29, 2016 at 3:52 pm

五月29，2016 3时：下午52 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-1298)

I asked the question about this code and WPF. After more reading and some head banging I got it to work. Now Im ready to move on.

我问了有关此代码和 WPF 的问题。经过更多的阅读和一些头部撞击，我让它开始工作。现在我准备继续前进。

Thanks.

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=1298#respond)

---

matt_p says: matt_p 说：
January 2, 2017 at 3:44 pm

一月 2， 2017 在 3：44 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-5909)

Hi, i skipped over your code and i found a few things confusing. First, your Model_View_Projection(MVP) matrix is calculated in the opposite order i am used to. I don't really know how the library you're using works but this seems a little bit confusing. As a follow up the project() function is even more confusing and i think its mainly due to what

嗨，我跳过了您的代码，发现一些令人困惑的事情。首先，您的 Model_View_Projection（MVP） 矩阵的计算顺序与我习惯的顺序相反。我真的不知道您使用的库是如何工作的，但这似乎有点令人困惑。作为后续，project（） 函数更加令人困惑，我认为它主要是由于什么

var point = Vector3.TransformCoordinate(coord, transMat);

var point = Vector3.TransformCoordinate（coord， transMat）;

is really doing and maybe this is connected with the confusion of the construction or your MVP matrix. At this point i would expect that this line transforms from object space -> camera space -> normalized screen space in the range from [-1,1]. You would then do the perspective divide and the transformation to raster space [0, screen_width] ... but nothing of this happens in your code. No perspective divide and a confusing mapping that suggests that point (the result of the confusing TransformCoordinate function) is within the range of [-0.5, 0.5]. I've never seen this before and i think its due to how Babylon works (never used it). Maybe you should clarify on that point because this seems to be very special in how Babylon works and is just very confusing if one would implement your code without Babylon maybe in another language.

真的在做，也许这与结构或 MVP 矩阵的混乱有关。在这一点上，我希望这条线从对象空间 ->相机空间 -> [-1，1] 范围内的标准化屏幕空间转换。然后，您将执行透视分割并变换为栅格空间 [0， screen_width] ... 但是在您的代码中不会发生任何情况。没有透视划分和令人困惑的映射，表明该点（令人困惑的变换坐标函数的结果）在 [-0.5， 0.5] 的范围内。我以前从未见过这个，我认为这是由于巴比伦的工作方式（从未使用过）。也许你应该澄清这一点，因为这在巴比伦的工作方式中似乎非常特别，如果有人在没有巴比伦的情况下用另一种语言实现你的代码，就会非常令人困惑。

best regards, Matt 最好的问候，马特

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=5909#respond)

Giovanni Passerello says:

塞雷洛 说：

January 3, 2017 at 2:40 pm

一月 3， 2017 在 2：40 下午 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-5950)

Hi Davrous, I am coding this in vb.net and have just installed SharpDX using NuGet onto Visual Studio 2015. My program is perfectly happy importing SharpDX, however when I try use Vector3 it does not recognise this… any suggestions? There seems to be no Vector3 on there?

嗨，Davrous，我正在用 vb.net 编码，并且刚刚使用NuGet将SharpDX安装到Visual Studio 2015上。我的程序非常高兴导入 SharpDX，但是当我尝试使用 Vector3 时，它无法识别这一点……有什么建议吗？那里似乎没有 Vector3?

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=5950#respond)

Brendan Burkhart says: 布伦丹·伯克哈特 说：

March 5, 2017 at 4:25 pm

三月5，2017 4时：下午25 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-8346)

I had this same issue, to solve it add "using SharpDX.Mathematics;" to your code, leave everything else as is.

我遇到了同样的问题，要解决它，请在您的代码中添加"使用 SharpDX.Mathematics;"，保持其他所有内容不变。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=8346#respond)

Jared says: 贾里德 说：
March 10, 2017 at 5:08 am
三月10，2017 5在：上午08 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-8525)

Hello, 你好
That did not solve the issue for me.

这并没有为我解决问题。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=8525#respond)

Brendan Burkhart says: 布伦丹·伯克哈特 说：
March 12, 2017 at 7:09 pm
三月12，2017 7时：下午09 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-8597)

You have used NuGet to install both the core SharpDX and SharpDX.Mathematics?

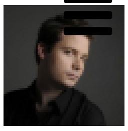
您是否使用 NuGet 安装核心 SharpDX 和 SharpDX.Math?

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=8597#respond)

Jared says: 贾里德 说：
March 19, 2017 at 6:10 pm
三月19，2017 6时：下午10 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-8732)

Hello, 你好
Thank you for your response. I initially used NuGet to download SharpDX. I attempted to use Mathematics extension from that and did not realize I had to also download the NuGet SharpDX.Mathematics first.

感谢您的回复。我最初使用NuGet下载SharpDX。我尝试从那里使用数学扩展，但没有意识到我还必须首先下载NuGet SharpDX.Mathematics。



Threezool (http://www.digitalzoolutions.com) says: 三虫说：
April 12, 2017 at 9:39 pm

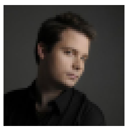
四月12，2017 9时：下午39 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-9916)

For any one interested in a updated sample code i just ported the project over to UWP and Visual Studio 2017 and published to GitHub:

对于任何对更新的示例代码感兴趣的人，我只是将项目移植到 UWP 和 Visual Studio 2017 并发布到 GitHub：

https://github.com/threezool/SoftEngineUWP (https://github.com/threezool/SoftEngineUWP)

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=9916#respond)



Threezool (http://www.digitalzoolutions.com) says: 三虫说：
April 12, 2017 at 10:57 pm

四月12，2017 10时：下午57 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-9918)

Added branches for the different parts

为不同部分添加了分支

Part 1: 篇一:
https://github.com/threezool/SoftEngineUWP/tree/Step1
(https://github.com/threezool/SoftEngineUWP/tree/Step1)

Part 2: 篇二:
https://github.com/threezool/SoftEngineUWP/tree/Step2
(https://github.com/threezool/SoftEngineUWP/tree/Step2)

Part 3: 篇三:
https://github.com/threezool/SoftEngineUWP/tree/Step3
(https://github.com/threezool/SoftEngineUWP/tree/Step3)

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=9918#respond)

Jia says: 贾 说：
March 26, 2018 at 1:59 pm

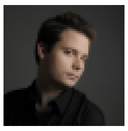
三月26，2018 1时：下午59 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-16854)

Hi, I've implemented your project with a back buffer of 640*480. But I found that the cube would be stretched into a cuboid. It is weird so I really wonder how to solve this problem.

嗨，我已经使用 640*480 的后台缓冲区实现了您的项目。但我发现立方体会被拉伸成长方体。这很奇怪，所以我真的想知道如何解决这个问题。

Thanks so much, 非常感谢，
Jia

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=16854#respond)



Threezool (http://www.digitalzoolutions.com) says: 三虫说：
April 12, 2017 at 11:00 pm

四月12，2017 11时：下午00 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-9919)

I also made branches for the first few parts of the guide.

我还为指南的前几部分制作了分支。

Part 1: 篇一:
https://github.com/threezool/SoftEngineUWP/tree/Step1
(https://github.com/threezool/SoftEngineUWP/tree/Step1)

Part 2: 篇二:
https://github.com/threezool/SoftEngineUWP/tree/Step2
(https://github.com/threezool/SoftEngineUWP/tree/Step2)

Part 3: 篇三:
https://github.com/threezool/SoftEngineUWP/tree/Step3
(https://github.com/threezool/SoftEngineUWP/tree/Step3)

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=9919#respond)

Jia says: 贾 说：
March 25, 2018 at 1:58 pm

三月25，2018 1时：下午58 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-16848)

thx so much! I am trying to implement it.

非常感谢！我正在尝试实施它。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=16848#respond)

Pingback: From zero to hero, creating WebVR experiences with Babylon.js on all platforms – David Rousset (https://www.davrous.com/2017/07/07/from-zero-to-hero-creating-webvr-experiences-with-babylon-js-on-all-platforms/)

Pingback：从零到英雄，在所有平台上使用巴比伦创造WebVR体验.js – David Rousset

Phil G says: 菲尔 G 说：
February 12, 2019 at 7:45 pm
二月12，2019 7时：下午45 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-21393)

Came across your tutorials about two thirds of the way through a very similar project but using vb.net (winforms). Though I took some different approaches e.g. I used Bresenham's line drawing algorithm for example. It was still very informative. Especially your description of the Z-buffer.

在一个非常相似的项目中遇到了你的教程，大约三分之二，但使用 vb.net（winforms）。虽然我采取了一些不同的方法，例如我使用了布雷森汉姆的线条绘制算法。它仍然非常翔实。尤其是您对 Z 缓冲区的描述。

So thank you. 所以谢谢你。

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=21393#respond)

David Rousset (https://www.davrous.com) says: 大卫·鲁塞特说：
February 12, 2019 at 7:49 pm
二月12，2019 7时：下午49 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-21394)

Great! Thanks for your feedback 🙂

伟大！感谢您的反馈 🙂

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=21394#respond)

Larry57 says: 拉里57 说：
December 7, 2019 at 8:29 pm
十二月7，2019 8时：下午29 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/#comment-32825)

Hi David, your tutorial is an absolute treasure.

嗨，大卫，您的教程绝对是宝藏。

I am trying to do the exact same thing as you, I mean, to learn 3D from bare C# using maths and Winforms GDI+. Except my work was based on the excellent site of Song Ho (http://www.songho.ca/opengl/index.html (http://www.songho.ca/opengl/index.html)) and his nice C++ projects.

我正在尝试做与您完全相同的事情，我的意思是，使用数学和Winforms GDI+从裸C#中学习3D。除了我的作品是基于Song Ho（http://www.songho.ca/opengl/index.html）的优秀网站和他的C++项目。

I managed to reach a high level of satisfaction seeing anithing running so smooth... Except for the rasterization part when trying to deal with lights. Suddenly, GDI polygons show their limits whenyou must modify inside.

看到动画运行如此流畅，我设法达到了很高的满意度......除了尝试处理灯光时的光栅化部分。突然，GDI 多边形在必须在内部修改时显示其限制。

The rasterisation part you explain is exactly what I was looking for, and drawing a line by myself directly in a byte[] buffer was something I did not even dare 🙂

您解释的光栅化部分正是我正在寻找的，我什至不敢🙂自己直接在 byte[] 缓冲区中画一条线

I cannot wait to try this backbuffer technique and implement direct byte[] drawing for triangles. Then, using TPL and peppering AggressiveInlines where it needs to.

我迫不及待地想尝试这种后台缓冲区技术并为三角形实现直接字节[]绘制。然后，使用TPL并在需要的地方加入AggressiveInline。

I cannot thank you enough for sharing your work !

我非常感谢您分享您的工作！

Reply 答 (https://www.davrous.com/2013/06/13/tutorial-series-learning-how-to-write-a-3d-soft-engine-from-scratch-in-c-typescript-or-javascript/?replytocom=32825#respond)

Pingback: Understanding Shaders, the secret sauce of 3D engines – David Rousset (https://www.davrous.com/2020/03/22/understanding-shaders-the-secret-sauce-of-3d-engines/)

Pingback：了解着色器，3D引擎的秘密武器——David Rousset

Pingback: 10 個GitHub庫開發者必看 [20200507-01] – jashliao部落格 (http://jashliao.eu/wordpress/2020/05/07/10-%e5%80%8bgithub%e5%ba%ab%e9%96%8b%e7%99%bc%e8%80%85%e5%bf%85%e7%9c%8b-20200507-01/)

Pingback： 10 个GitHub库开发者必看 [20200507-01] – jashliao部落格

Pingback: Write Your Own 3D Engine (0) – Yih Horng (http://www.yhorng.com/blog/?p=52)

Pingback： 编写自己的 3D 引擎 （0） – Yih Horng

Pingback: 3D Software Rendering on the GBA | Brain Baking (https://brainbaking.com/post/2020/07/3d-software-rendering-on-gba/)

回调：大湾区的3D软件渲染 |大脑烘焙

Leave a Reply 留言

Your email address will not be published. Required fields are marked *

您的电子邮件地址将不会发布。必填字段标记为 *

Comment * 评论*

Name * 名字*

Email * 电子邮件*

Website 网站

Post Comment