《前端图形学启蒙课》

yuanzhijia@yidengxuetang.com

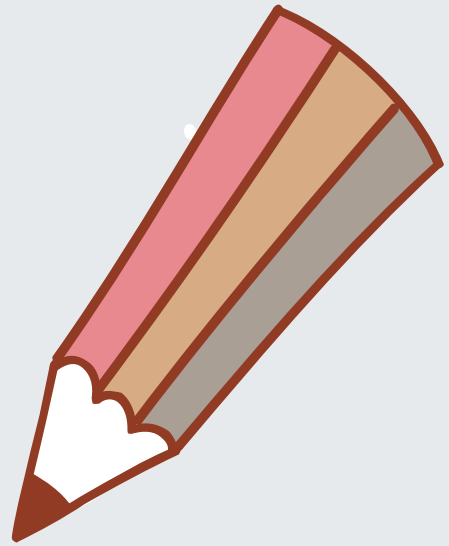
1 走进图形学工程师的世界

2 WebGL核心概念解读

3 10分钟快速上手ThreeJS

4 前端跨界之开发AR VR

5 WebGL大型项目的应用



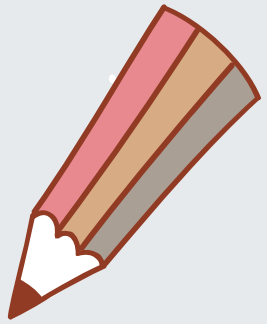
目录

contents

走进图形学工程师的世界

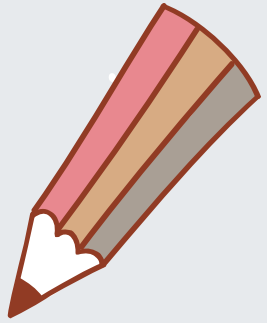
了解前端图形学工程师的工作范畴

抛出一些问题



① 传说中的图形开发工程师，B格好像很高，很难的样子，我数学能做吗?

② 本周的预习课里会有很多知识点自己写没思路怎么办，是不是就无缘图形学了?



① 很多陌生的属性(vertexAttrib4f、uniform4f)，或者奇怪的方法命名（gl.clear()）好多好长脑袋不够用！

② Three.js！=WebGL！=3D,其实不懂的时候真的觉得很难，懂了之后就非常简单。

来点和普通网页不一样的

数据可视化

ECharts、Highcharts、D3.js

跨界AR和VR

Reacr-VR/AR、ar.js、renderloop

游戏开发

Cocos2d-JS、Egret、

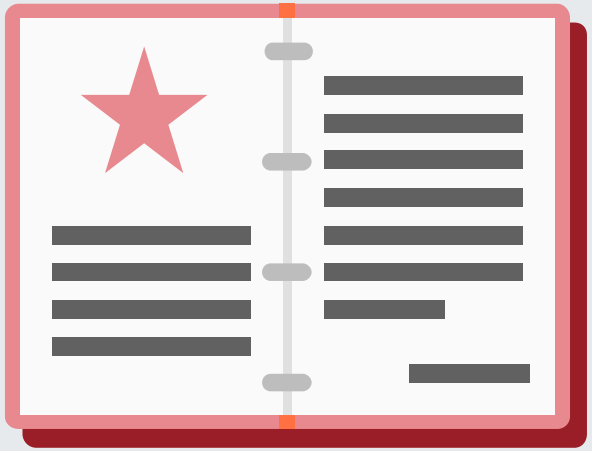
CreateJS、Phaser

3D开发

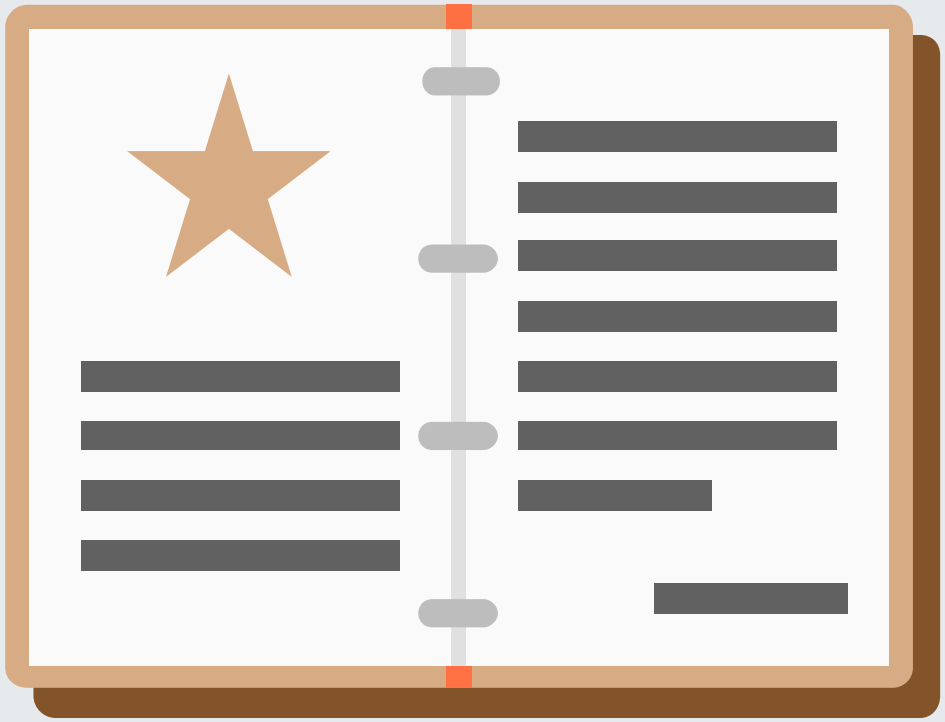
Three.js、xeogl、cesiumo、

babylonjs

基本的数学知识

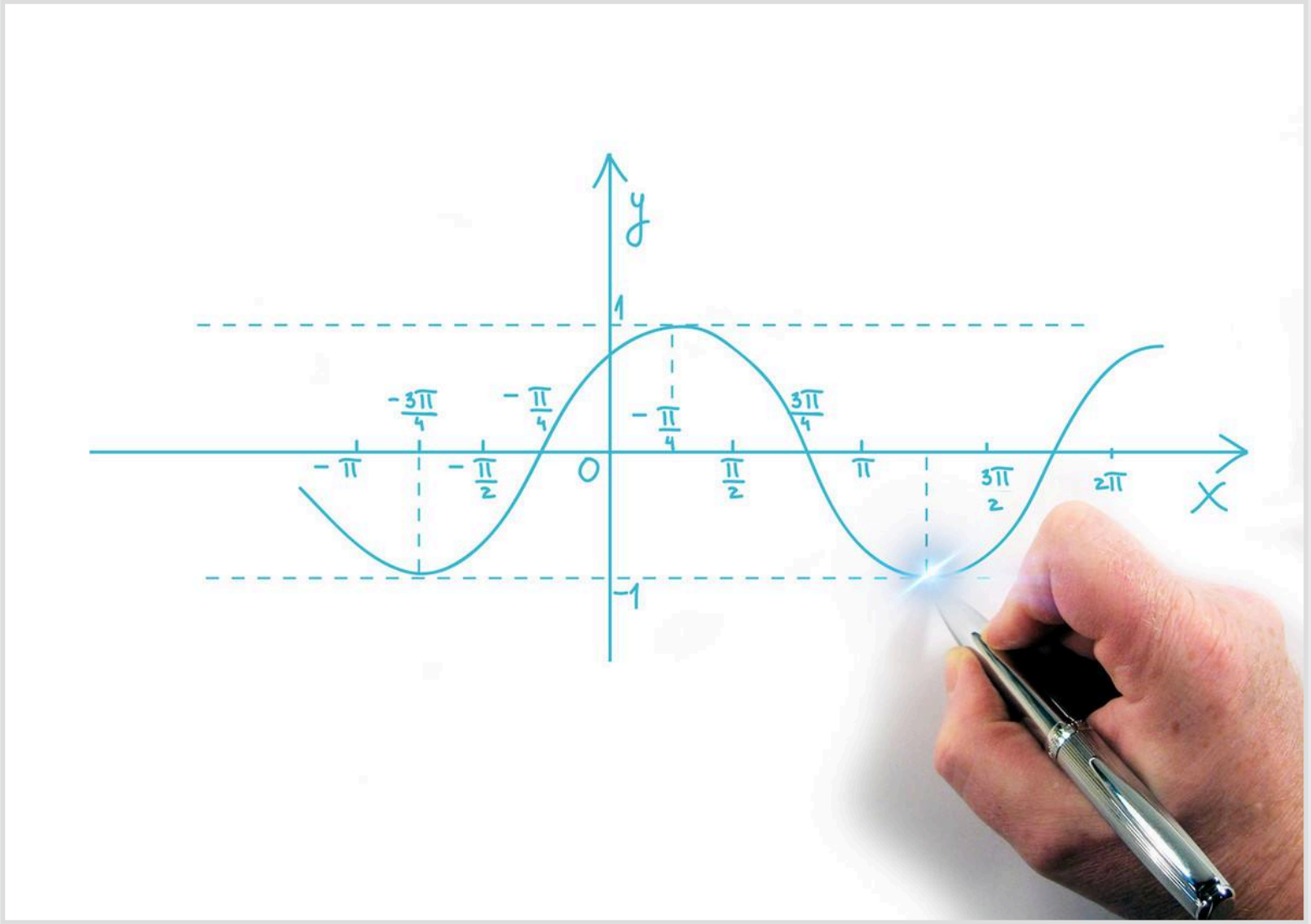


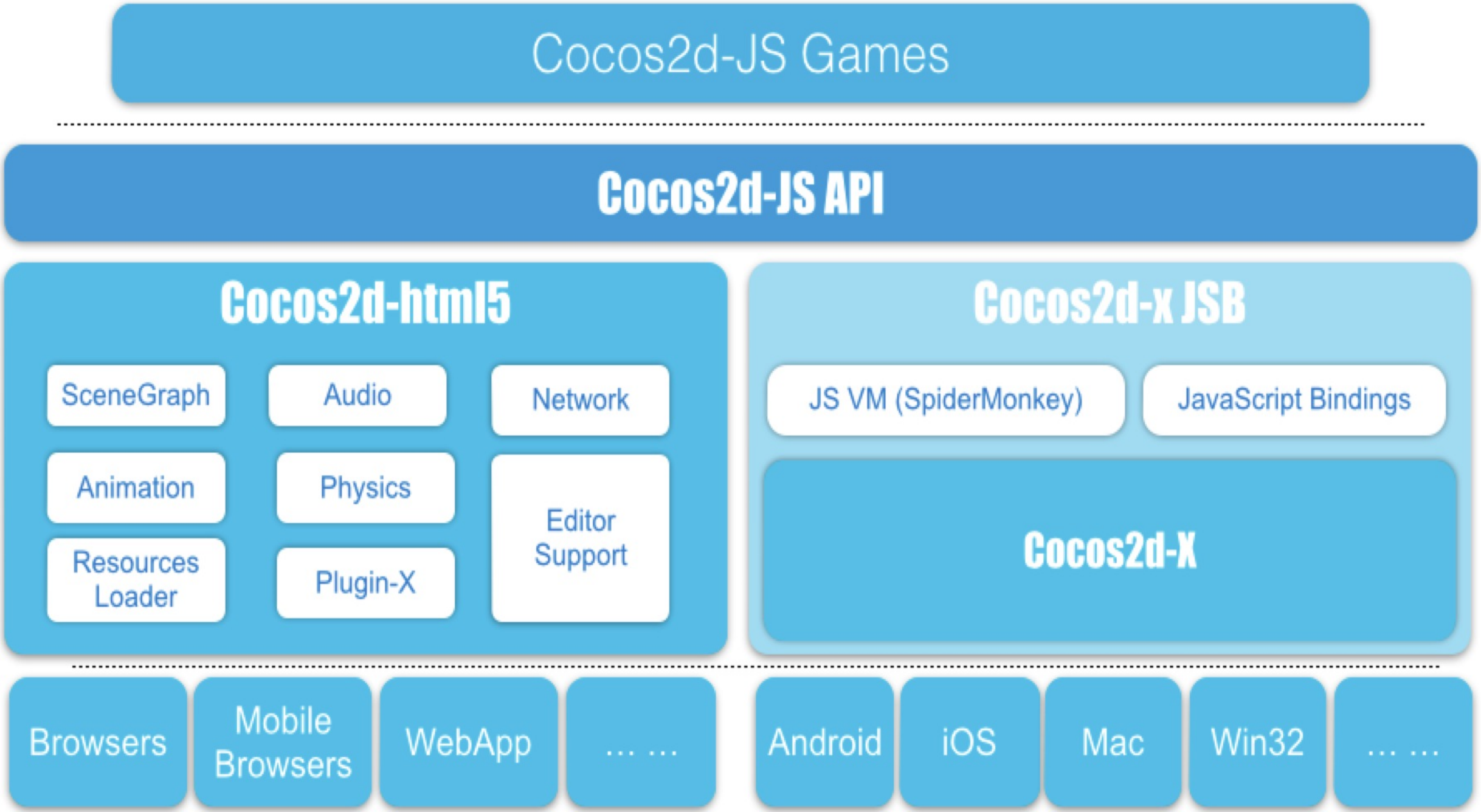
高中物理
高中数学



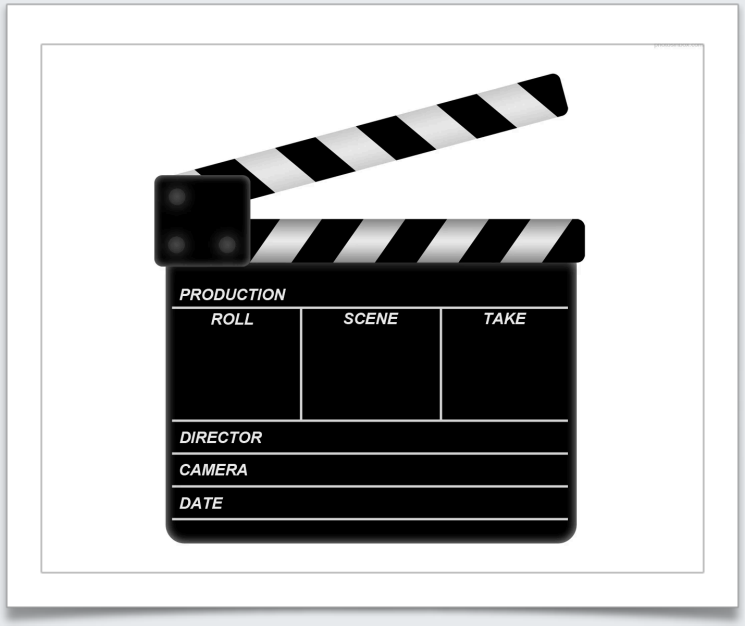
大学物理
大学数学

实战自由落体并受空气阻力




Cocos2d-JS Games
Cocos2d-JS API
Cocos2d-html5
SceneGraph
Audio
Network
Animation
Physics
Editor Support
Resources Loader
Plugin-X
Cocos2d-x JSB
JS VM (SpiderMonkey)
JavaScript Bindings
Cocos2d-X
Browsers
Mobile Browsers
WebApp
… …
Android
iOS
Mac
Win32
… …

2D环境中必要的因素



WebGL核心概念解读

从介绍到一些核心的入门知识

更深入的理解WebGL/OpenGL/DX到底是干嘛的?

请我们生活的世界是三维的，但显示器是

二维的

数学家们就YY了一套三维的数学模型来模

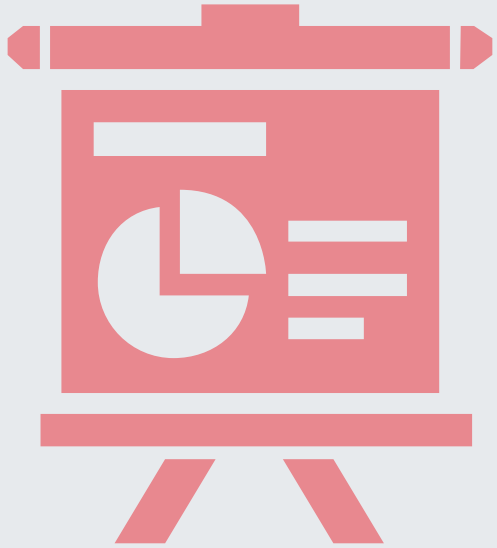
拟数学（程序）界的三维世界，但是坑爹

的显示器始终是二维的!

数学家们想起了厉害的画家，就算在二维的纸上画的东西也和三维的一样。

然后程序员就把纸上的画搬到了二维的显示器上。

继续理解



一些基本概念

“

- 显示器不认识数学家YY的数学模型，它只认像素图。
- 图形工程师就是把数学家YY的数学模型给转化成图片扔给显示器显示出来。
- 要做好上面的事，学习WebGL=学数学家YY的数学模型+怎么把数学模型转化成图片。
- OpenGL/Dx 规定与封装了把数学模型转换成图片的过程
- 数学不好能做WebGL！但当不了图形开发工程师！

它们之间的关系八卦描述

WebGL、OpenGL、JavaScript、C、DX、DirectX、C++、iOS、Android

WebGL就是OpenGL和JavaScript生的残疾儿子（目前来讲比较残疾）

OpenGL就是一个英文公司封装写的一套C系API(认为自己，更灵活，性能比DX好，应该当各种平台（除了xbox）图形处理方面的标准协议)

DirectX是微软爸爸封装写的一套C++系API(认为自己，封装的更彻底简单暴力，并且性能也不比OpenGL差，而且更适合做游戏)

微软爸爸仗着自己高大威猛，想用DX取代了OpenGL。然后OpenGL一个人干不过，就叫来了各种浏览器、ios、安卓、ps3联合震压DX（PC、XBOX）

WebGL怎么把数学模型转成图片？

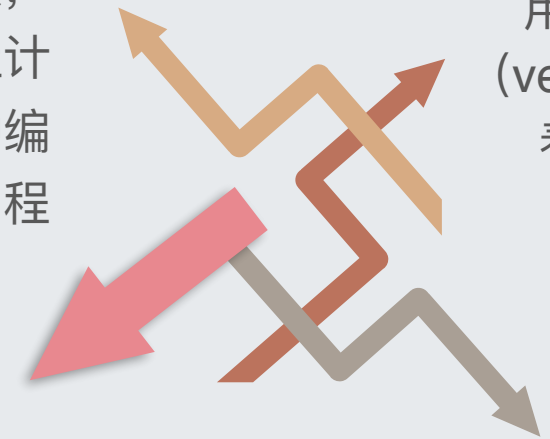
- 1.OpenGL中有固定渲染管线和可编程渲染管线。
- 2.WebGL（残疾的儿子）目前只有可编程渲染管线
- 3.固定渲染管线：就是一套把数学模型转成图片的套路—模型坐标转换(M)、视图坐标转换(V)、投影坐标转换(P)简称MVP套路。

WebGL

- 4.模型坐标转换：模型的基准点（原点）、模型的大小、模型旋转角度等。
- 5.视图坐标转换：从哪个方向、角度观看这个模型
- 6.投影坐标转换：离的越近的呈现出来的图像就应该越大，越远越小。

WebGL怎么把数学模型转成图片？

1.可编程渲染管线：固定的MVP套路计算机会自己跑，但有的时候，其中某些套路不想让计算机自动处理，手动编写某个套路就是可编程渲染管线。



2.着色器（shader):就是帮助我们做可编程渲染管线的工具。WebGL常用的有顶点着色器(vertex shader)、片元着色器(fragment shader)。

3.顶点着色器：顾名思义它能处理顶点坐标、大小等（矩阵计算后的结果），能够把数学坐标光栅化。

4.片元着色器：能够接收光栅化数据并加以处理使其显示到屏幕上 (光栅化数据包含了像素的位置、颜色等信息)

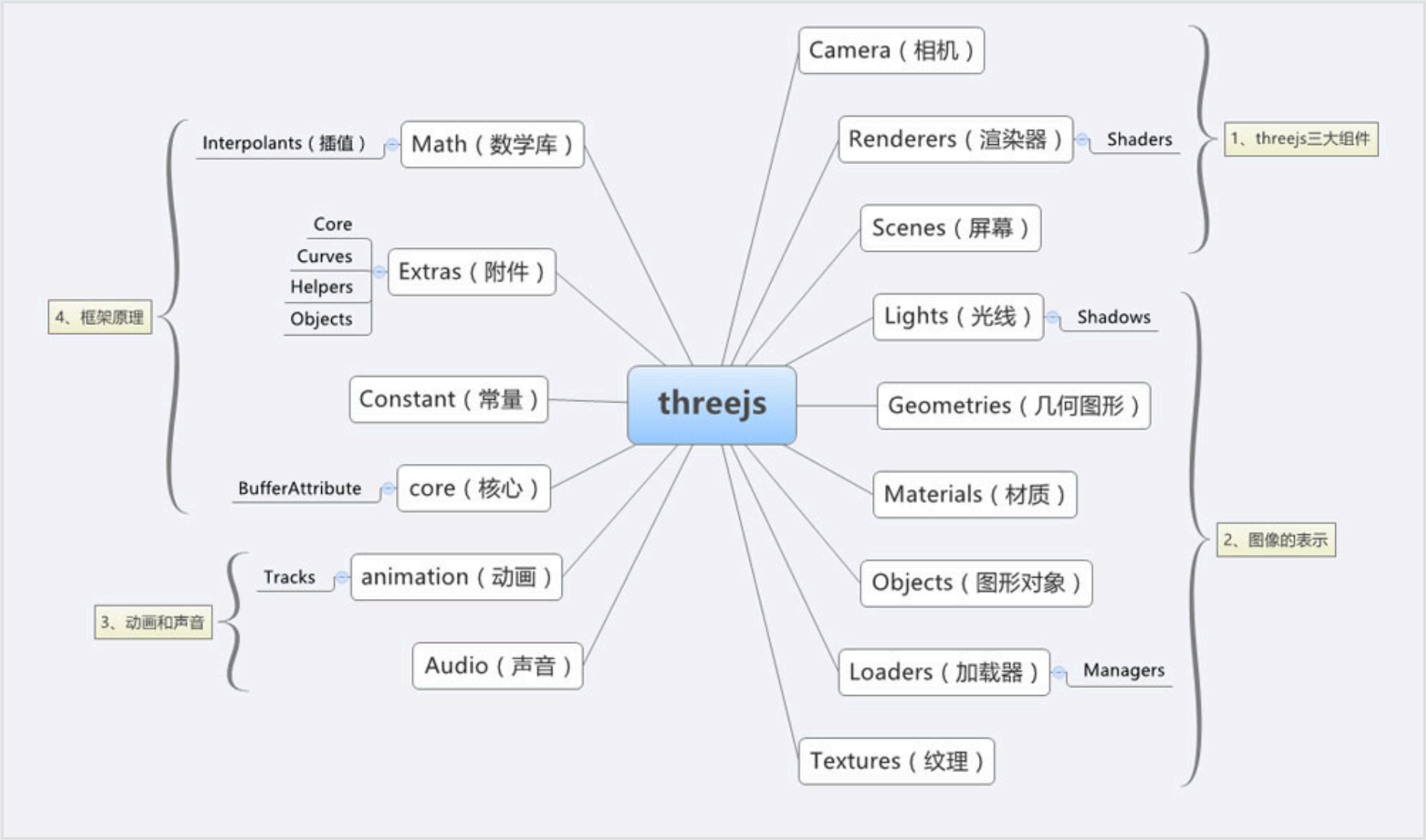
光栅化：光栅化就是把顶点数据转换为片元的过程。

A. 片元中的每一个元素对应于帧缓冲区中的一个像素。 光栅化其实是一种将几何图元变为二维图像的过程。该过程包含了两部分的工作。第一部分工作：决定窗口坐标中的哪些整型栅格区域被基本图元占用；第二部分工作：分配一个颜色值和一个深度值到各个区域。光栅化过程产生的是片元。把物体的数学描述以及与物体相关的颜色信息转换为屏幕上用于对应位置的像素及用于填充像素的颜色，这个过程称为光栅化，这是一个将模拟信号转化为离散信号的过程。

10分钟快速上手ThreeJS

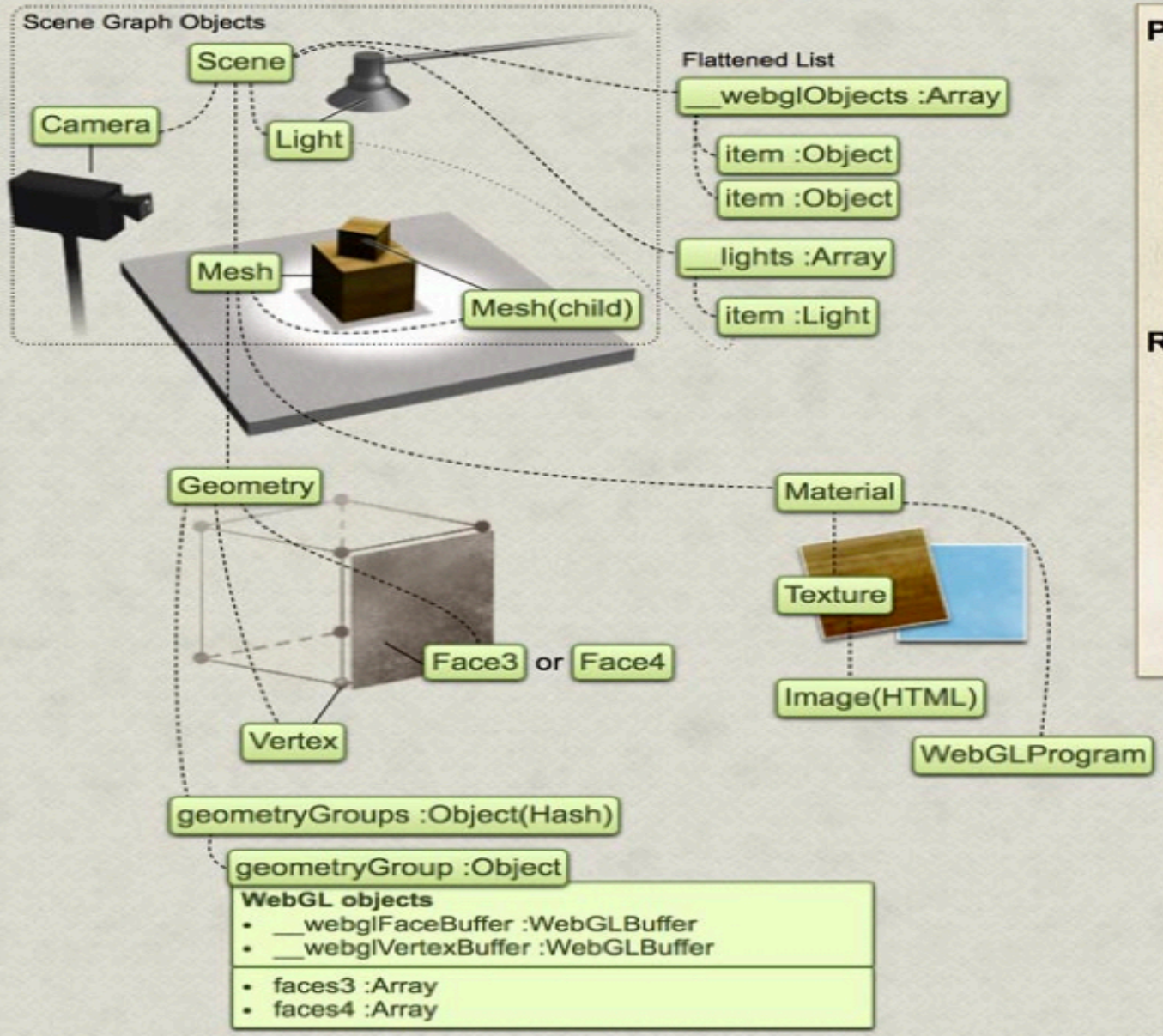
让我们化简为繁快速上手Three

Three.js目录结构



Three.JS Walking Map

Structure of Core Objects



Basic Rendering Sequence

Preparation Phase

1. [Setup flattened list] For each scene graph object:
    1. Init matrices.
    2. Create geometry groups corresponding to face materials.
    3. [Setup buffers] For each geometry group object:
        1. Create WebGL buffers.
        2. Create Non-WebGL buffers.
        3. Add an item to flattened list.
2. Update buffers (Vertex buffer, Index buffer...).

Rendering Phase

1. Update matrices on all of the descendants of scene.
2. Set up camera matrices.
3. [Setup-loop] For each flattened list item:
    1. Set up matrix.
    2. Pick materials.
4. [Draw-loop] For each flattened list item:
    1. Set up shader programs(and textures).
    2. Prepare buffers.
    3. Draw primitives

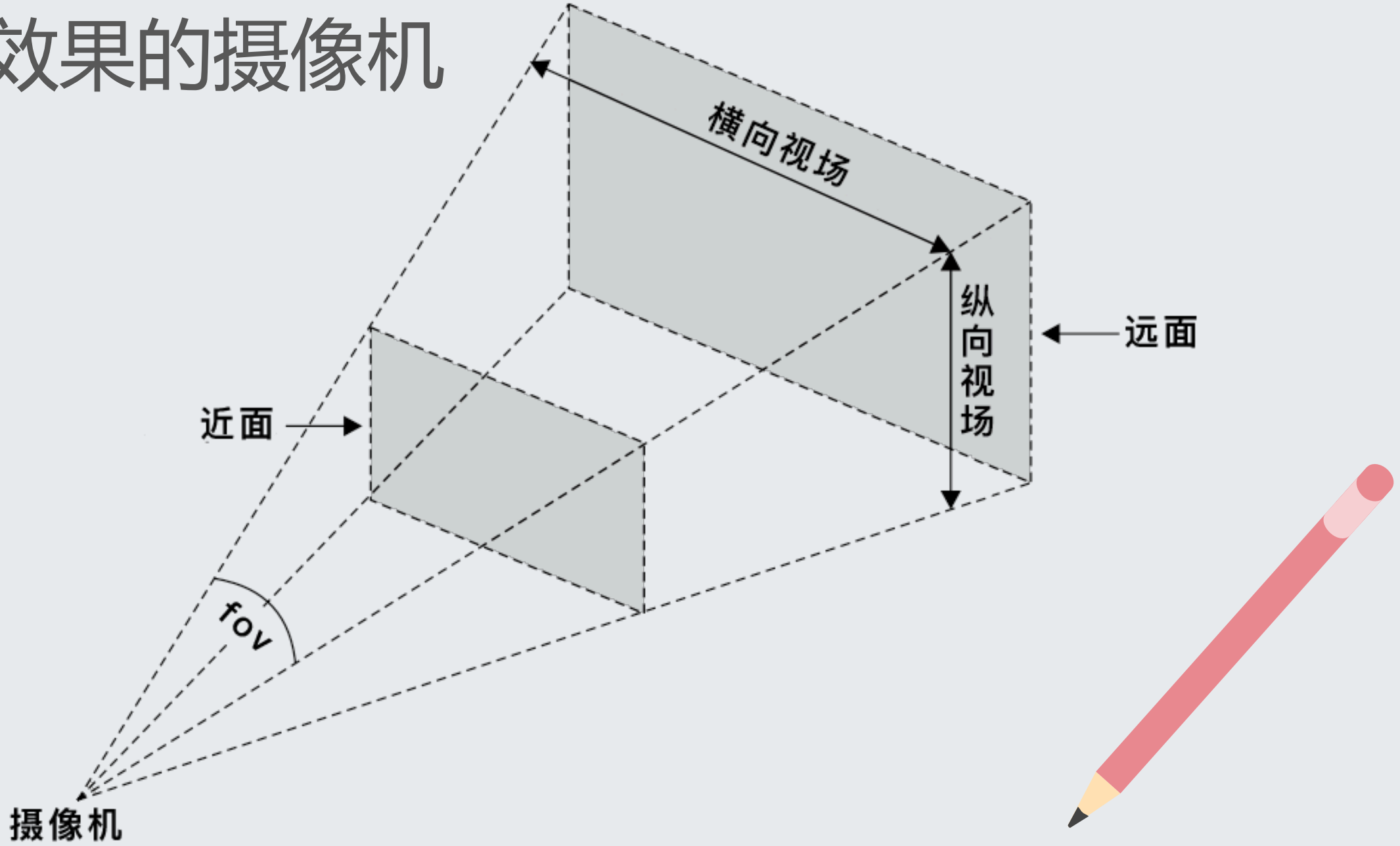
3D的基础概念

纹理即“纹路”，每个物体表面上不同的样子，譬如说木头的木纹状。

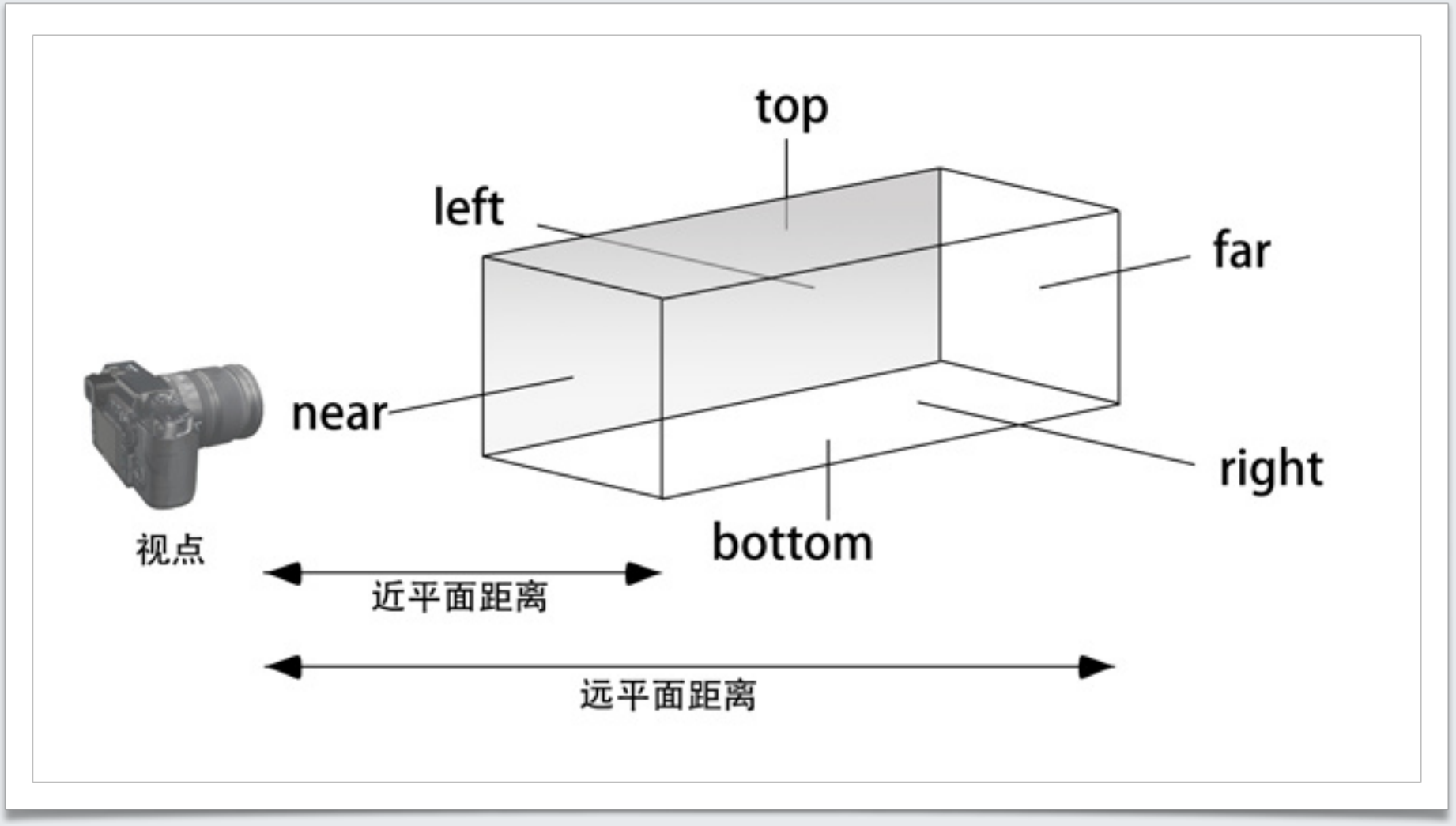
贴图是图，最简单的形式是ps之类的软件做出来的一张图，这些图在3D中用来贴到物体的表面，用来表现物体的“纹理”。

材质主要是用来表现物体对光的交互（反射、折射等）性质的。譬如金属对光的反射和毛毯对光的反射性质完全不一样，那么对3D程序来说，这样的差别就通过材质这个属性来计算出不同的颜色。

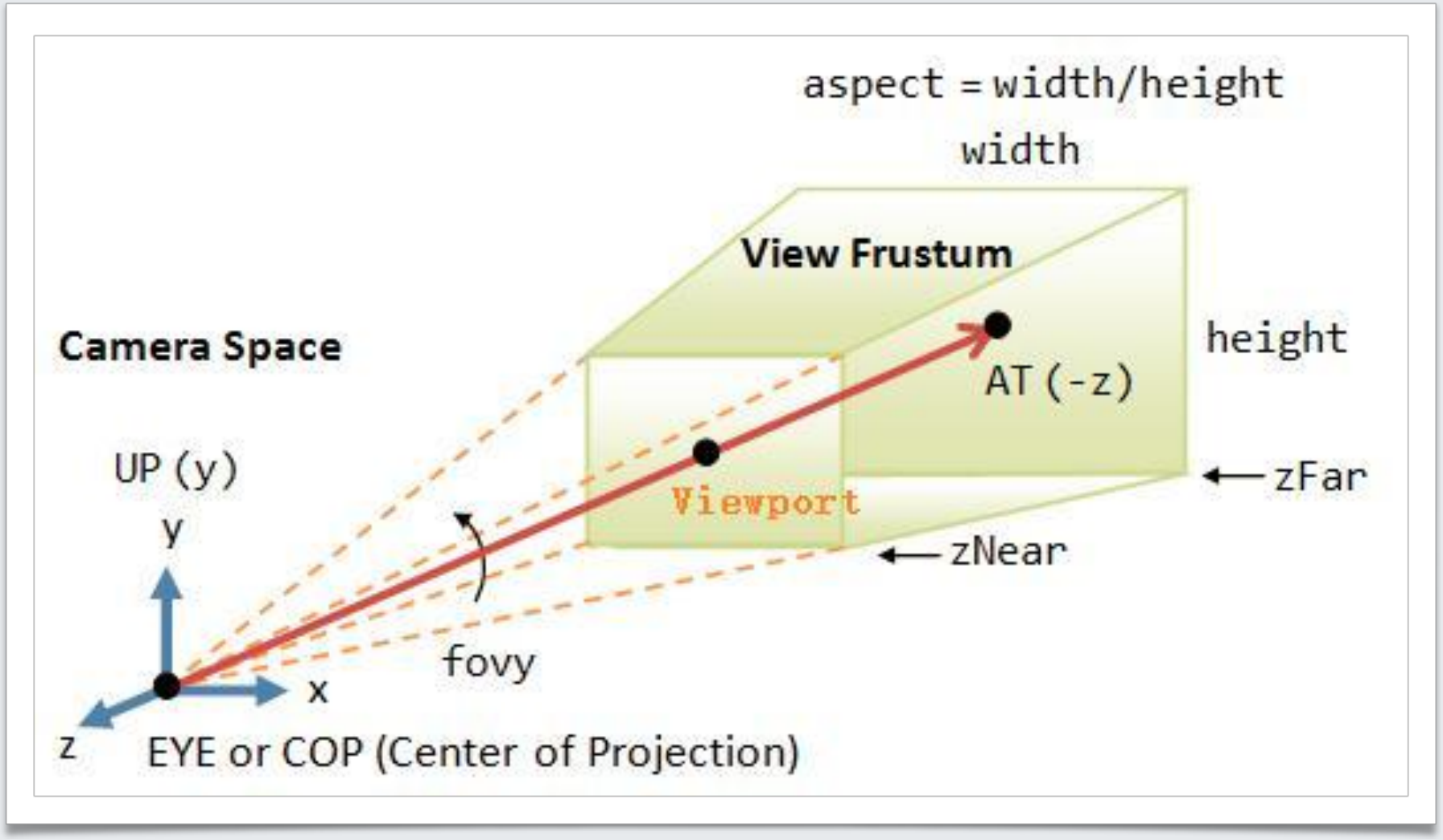
透视效果的摄像机

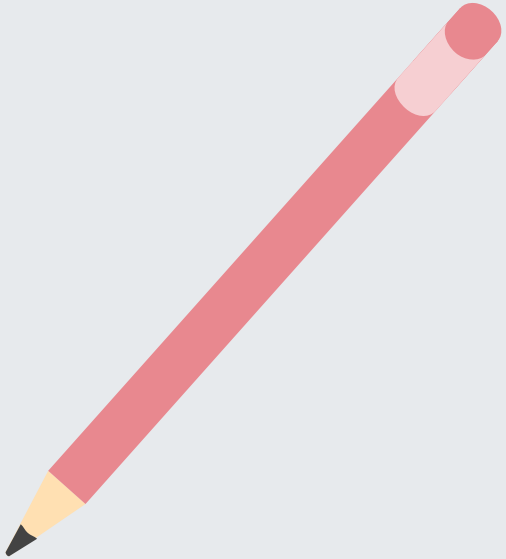


透视效果的摄像机



透视效果的摄像机

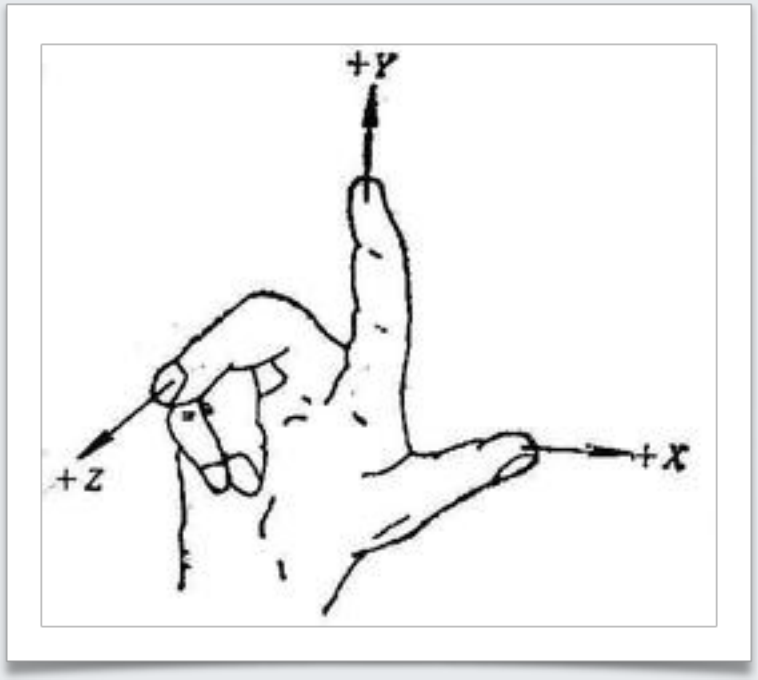

aspect = width/height
width
View Frustum
Camera Space
height
AT (-z)
UP (y)
y
zFar
Viewport
zNear
fovy
x
z
EYE or COP (Center of Projection)



坐标系的位置和指向？

“

坐标系的原点在画布中心（canvas.width / 2, canvas.height / 2）



”

“

通过 Three.js 提供的 THREE.AxisHelper() 辅助方法将坐标系可视化。

”

THREE.PerspectiveCamera(fov, aspect, near, far)

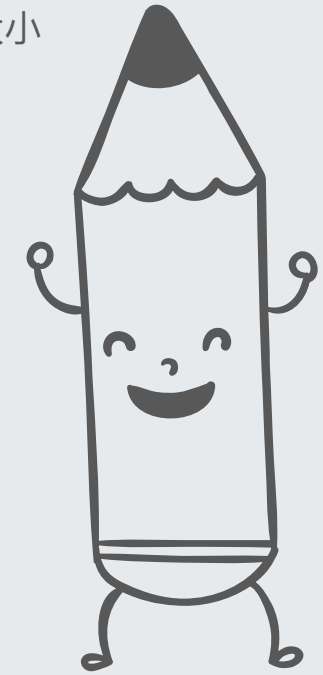
视场，即摄像机能看到的视野。比如，人类有接近 180 度的视场，而有些鸟类有接近 360 度的视场。但是由于计算机不能完全显示我们能够所看到的景象，所以一般会选择一块较小的区域。对于游戏而言，视场大小通常为 60 ~ 90 度。
推荐默认值为：50

fov

指定渲染结果的横向尺寸和纵向尺寸的比值。在我们的示例中，由于使用窗口作为输出界面，所有使用的是窗口的长宽比。

推荐默认值：window.innerWidth / window.innerHeight

aspect



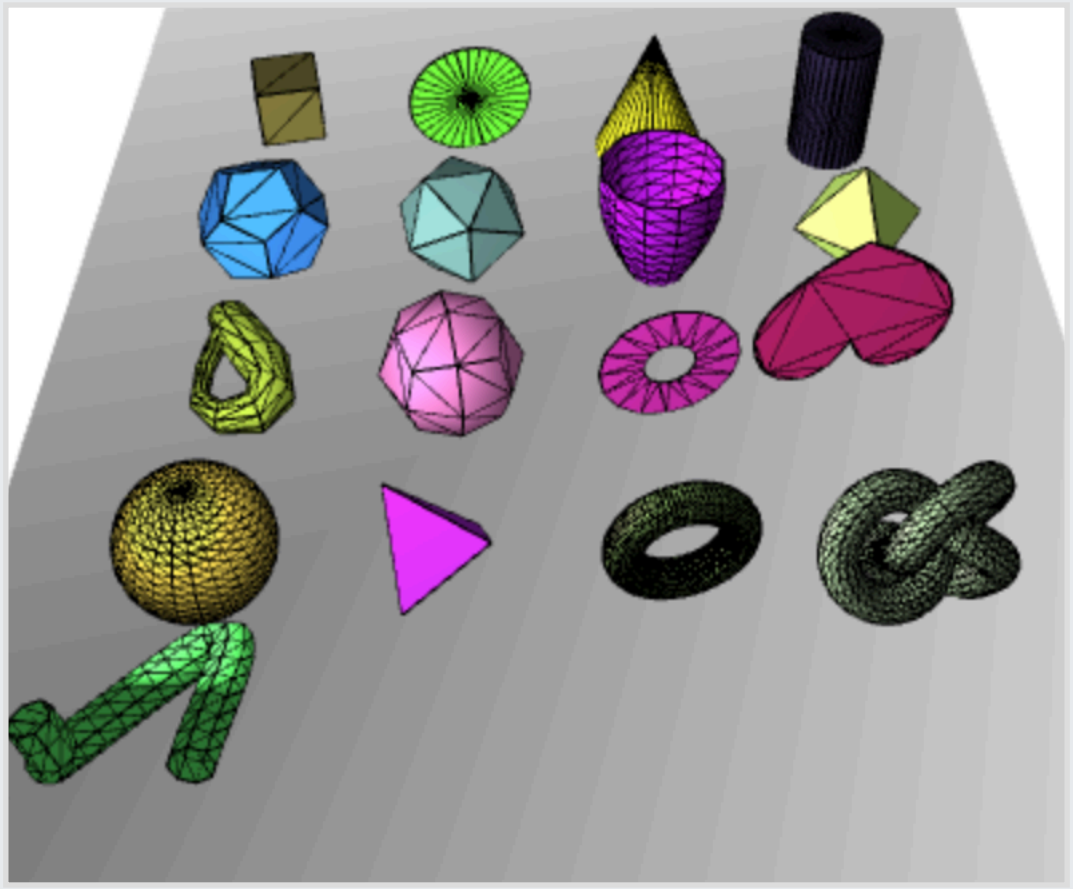
指定从距离摄像机多近的距离开始渲染。推荐默认值：0.1

near

指定摄像机从它所处的位置开始能看到多远。若过小，那么场景中的远处不会被渲染；若过大，可能会影响性能。
推荐默认值：1000

far

Mesh



Mesh 好比一个包装工

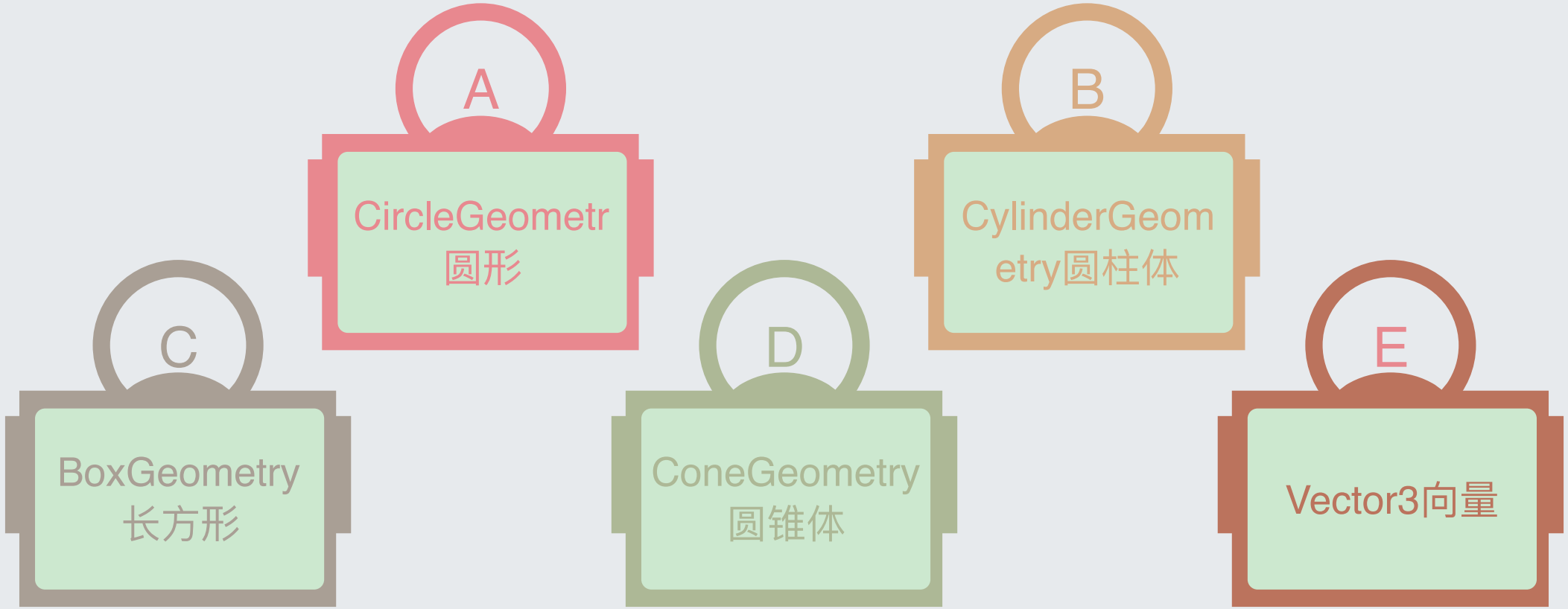
将『可视化的材质』粘合在一个『数学世界里的几何体』上，形成一个『可添加到场景的对象』

创建的材质和几何体可以多次使用

还有 Points（点集）、Line（线/虚线）等

Mesh

网格就是一系列的多边形组成的，三角形或者四边形，网格一般由顶点来描绘，我们看见的三维开发的模型就是由一系列的点组成的。



Mesh

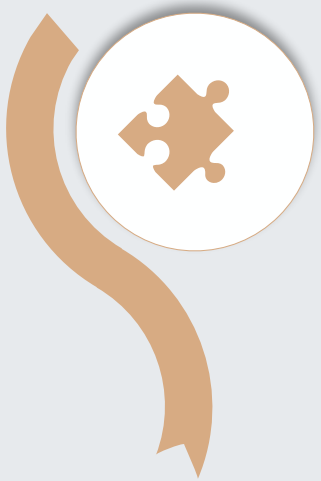
网格就是一系列的多边形组成的，三角形或者四边形，网格一般由顶点来描绘，我们看见的三维开发的模型就是由一系列的点组成的。

BoxGeometry（长方体）	CircleGeometry（圆形）	ConeGeometry（圆锥体）	CylinderGeometry（圆柱体）
DodecahedronGeometry（十二面体）	IcosahedronGeometry（二十面体）	LatheGeometry（让任意曲线绕 y 轴旋转生成一个形状，如花瓶）	OctahedronGeometry（八面体）
ParametricGeometry（根据参数生成形状）	PolyhedronGeometry（多面体）	RingGeometry（环形）	ShapeGeometry（二维形状）
SphereGeometry（球体）	TetrahedronGeometry（四面体）	TorusGeometry（圆环体）	TorusKnotGeometry（换面纽结体）
TubeGeometry（管道）	\	\	\

粒子 一直面向摄像机（无论你旋转摄像机还是设置粒子的 rotation 属性）

THREE.Sprite(material)

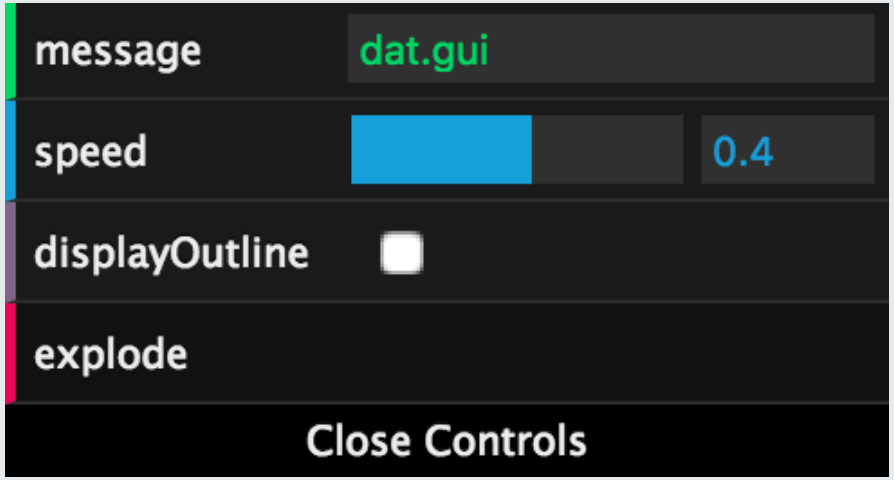
可以加载图片作为粒子的纹理



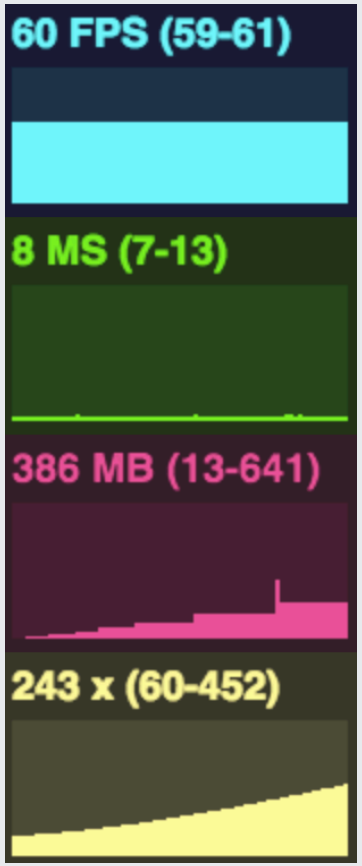
THREE.Points(geometry, material)

创建纯点作为粒子

需要的插件

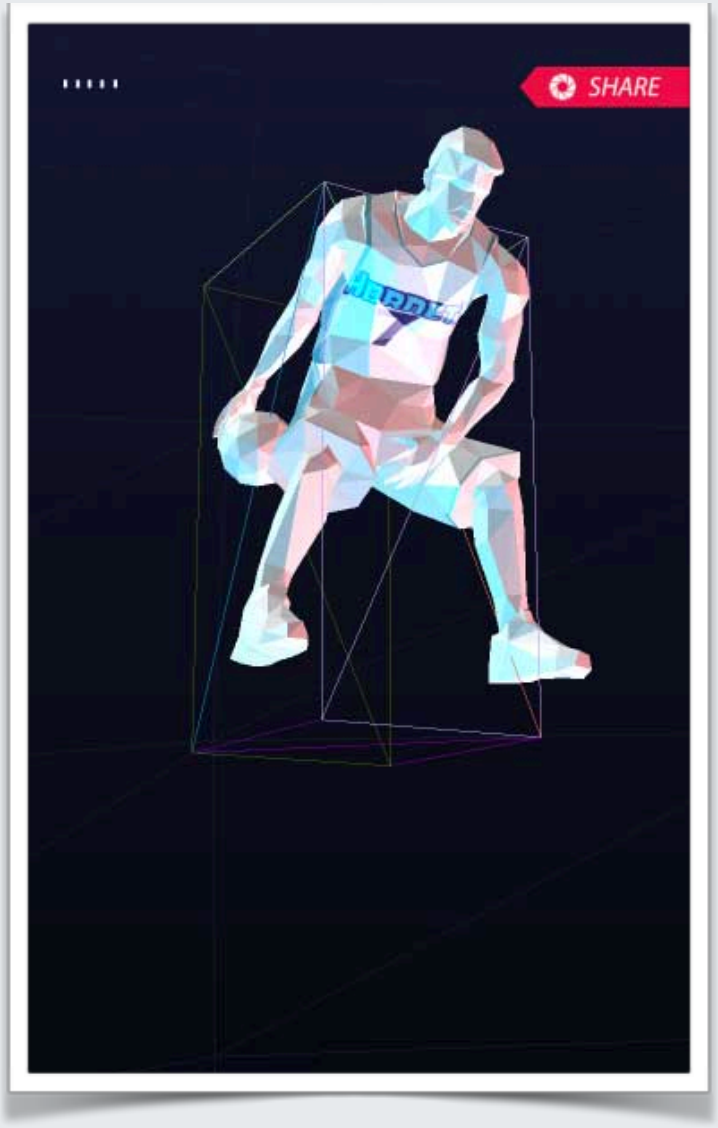


dat.GUI 提供了可视化调参的面板，对参数调整的操作提供了极大的便利。



stats.js 帧率、每帧的渲染时间、内存占用量、用户自定义

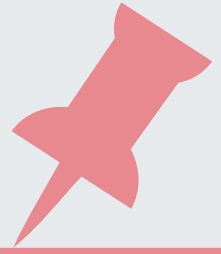
操作模型



前端跨界之开发AR VR

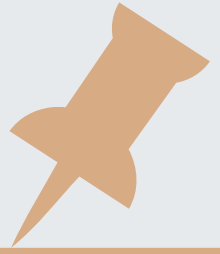
走进AR VR的开发圈子

选择合适的开发工具



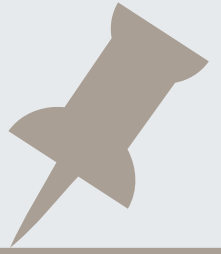
大部分的VR盒子都是通过光学透镜把手机屏幕的画面转变为VR画面，使用户享有沉浸式的体验。这一产品的代表作为谷歌Cardboard，这个只需要5元。建议买国产货。

VR盒子



VR一体机是指那些具备了独立处理器的VR设备，他们不再通过手机或电脑的处理器运行，具有独立输出输入能力，自成一体，故称一体机。推荐一款柔宇。

VR一体机



VR头显是目前最顶级的VR设备，在这个领域里有HTC Vive和Oculus Rift配合一部高配置的电脑主机（最低配置为CPU：I5 显卡：N卡970）,但是你想要享有顶级的VR体验，花个六七千是必须的了。

VR头盔

暴风VR

零镜小白VR

我用过的

WebGL大型项目的应用

应用在大型项目的细枝末节

性能优化点

图形学里面有个很重要的概念叫“one draw all”一次绘制，也就是说调用绘图api的次数越少，性能越高。

开启真正的网页多线程时代

控制模型的大小，对模型进行压缩。DRACO

谢 谢 列 位