1 attribute 属性变量只允许使用在顶点着色器

2. uniform 制服变量 既能用于顶点着色器，又能用于片段着色器

3 varying ⇒ 用来传输顶点着色器到片段着色器的数据

4. 精度 lowp、mediump、highp

5. 顶点着色器负责定位 3D中该网格的每个顶点，

```
void main(){
  gl_Position = projectionMatrix · modelViewMatrix * vec4(position, 1.0);
}
```

相机　　一个顶点的初始位置

如果有100个顶点，则上面的函数会执行100次
保证每一个顶点都会被放在正确的位置

2) 边段着色器 ⇒ ① 将顶点着色器的结果效果分成多个片段
② 对片段进行着色

```
void main(){
  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}
```

R　G　b

// 存储了个颜色的位置

alpha通道最主要的用途是存储和成立选用范围。
Alpha通道 [A (Opacity)]

6. 要想将图像完整的映射到图像表面，
即将图像完整的放在平面上，我们可以尝试将平面设置为纹理的第二修数，但是 three js 并没有为我们提供该接口，
但是我们可以使用 uv 坐标来尝试实现
three js 以 uv 为顶点着色器提供了一个属性，我们需要将该坐标的值从顶点着色器传递到片段着色器
我们需要使用方差存储限定符
首先，我们需要在顶点着色器创建一个 vec2 变量
正常情况下，贴图和uv坐标的操作都要放在着色器里，
但是目前的demo中，贴图，uv坐标，着色器是分离的，我不知道如何优化当前代码

```
const vertexShader =
varying vec2 vUv;
  ...
void main(){
  vUv = uv;
  ...
};
const fragmentShader =
varying vec2 vUv;
  ...
void main(){
  vec4 textureColor = texture2D(textureTooth, vUv + 0.1);
  ...
};
```

这 vUv 就是顶点着色器留给片段着色器的
⇓
再次印证了：varying 是负责顶点着色器与片段着色器进行交流时用到的。

7. 关于 Cannon.js ⇒ 用来模拟重力、摩擦、碰撞、惯性等。
1) 模拟个物理世界，引入牛顿等几大定律以及其它的一些定律
npm install cannon-es
$g = 9.81 m/s^2$
这部分用不到，跳过

8 如何在点击时创建1个小球？（6节）
一个相当棘手的一点：3js没有内置函数可以将鼠标位置 $\xrightarrow{转化}$ 3js中的世界坐标系
寻求 stackOverflow 的帮助
解决办法：
① 创建一个必须面向相机的隐形平面（每次鼠标移动时都需要执行此操作）
② 我们需要在射线与平面的交点调用 set 方法（设置平面的方向 → 用个单位法线；设置1个点）
⇓　　⇓
⇐　　世界坐标的原点

画个图来具象化：

③ 我们需要在射线与平面的交点处创建个小球

这涉及到 raycaster 类的知识点，
mouse：鼠标在浏览器中的坐标（Vector2）⇒
intersectionPoint：射线与平面的交点（Vector3）
planeNormal：平面法线（Vector3）
plane：平面
raycaster：光射投射器

www.baidu.com
鼠标只要移动就会触发

```
const mouse = new THREE.Vector2();
const intersectionPoint = new THREE.Vector3();
const planeNormal = new THREE.Vector3();
const plane = new THREE.Plane();
const raycaster = new THREE.Raycaster();
```

```
window.addEventListener('mousemove', function(e){
  mouse.x = (e.clientX / window.innerWidth)*2 - 1,
  mouse.y = -(e.clientY / window.innerHeight)*2 + 1,
  planeNormal.copy(camera.position).normalize();
  plane.setFromNormalAndCoplanarPoint(planeNormal, scene.position);
  raycaster.setFromCamera(mouse, camera);
  raycaster.ray.intersectPlane(plane, intersectionPoint);
});
```

→ 平面法向量复制相机的位置并进行规一化处理 ⇒ 生成单位向量
→ 平面实例调用set方法，参数（法线，坐标）创建1个平面
→ 创建射线
→ 与平面相交的方法（args：平面，存储交点坐标的变量）

使用set创建平面，就是隐形的
如果你使用了 planeGeometry 与 planeStandardMeterial 则是可视的

平面与射线的交点

补充：Mat 是矩阵的意思：网格 = 几何体 + 材质 ⇒ 材质可以看成1个矩阵

```
window.addEventListener('click', function(e){
  const sphereGeo = new THREE.SphereGeometry(0.125, 30, 30),
  const sphereMat = new THREE.MeshStandardMaterial({
    color: 0xFFEA00,
    metalness: 0,
    roughness: 0,
  });
  const sphereMesh = new THREE.Mesh(sphereGeo, sphereMat);
  scene.add(sphereMesh);
  sphereMesh.position.copy(intersectionPoint),
});
```

最后，再补个流程图：

① 鼠标移动事件
鼠标的位置始终有个隐形的平面（set方法），并不断从相机的位置发射射线记录隐形平面与射线的交点坐标

set方法是key，如果不知道，是无法实现这个效果的。

② 鼠标点击事件
在隐形平面与射线交点的位置创建1个小球模型。