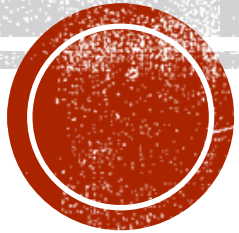
SAY HELLO TO CANVAS

—— Canvas 从入门到放弃



目录

- Canvas 2D
  - 上手Canvas 2D – Danmaku
  - 2D 贴图~
- Canvas 3D
  - 基础理论
  - Three.js – 3D 直播例子
- Live 2D – 其实是3D的2D

上手 CANVAS 2D

- this.canvascontainer = document.createElement('canvas');
  - // 创建一个Canvas 节点 等同于 <canvas> </canvas>
- context = danmaku.canvascontainer.getContext('2d');
  - // 告诉浏览器 这是一个2D画布，并拿到画布~
- requestAnimationFrame(render);
  - // 这个货就是浏览器在渲染下一帧时需要干点嘛。最高FPS 60。如果代码过于复杂，FPS会降低，火狐浏览器（由于渲染效率实在太差了）这个值本身就不高，随便来点context.xxxx就可能降低到 10-15. 人眼可以明显看到卡顿。所以火狐下慎用canvas. 实在要用，必须注意算法复杂度。

DANMAKU CANVAS 渲染

- render = function () {
- context.clearRect(0, 0, danmaku.width, danmaku.height);
  - 清屏！
- danmaku.runlines.forEach(function(runline) {
- runline.forEach(function(comment) {
- if (comment.type === 1 && comment.pool === 0) {
- _renderText(comment, now, shiftLines);
  - 渲染每条弹幕
- }});}); }

DANMAKU CANVAS 渲染

- function _renderText(comment, now, shiftLines) {
- 	context.font = comment.fontsize + 'px ' + font;
- ……
- context.fillStyle = comment.color;
- context.strokeStyle = comment.shadowColor;
- context.strokeText(comment.content, x, y);
- context.fillText(comment.content, x, y);
- }

目录

2D游戏基础-贴图

- 图层顺序贴图
  - function render() {
  - context.clearRect(0, 0, 600, 250);
  - renderMap(); // 一般会是最底下一层
  - renderMan();
  - requestAnimationFrame(render);
  - }

2D游戏基础-贴图

- Map
- var map = [
- [0,0,0,0,0],
- [0,1,1,1,0],
- [0,1,1,1,0],
- [0,1,1,1,0],
- [0,0,0,0,0]
-];

2D游戏基础-贴图

```
function renderMap() {
  map.forEach(function(val, row) {
    val.forEach(function(value, col){
      var x = row * 100 + (col % 2) * 50;
      var y = col * 25 + 100;
      context.drawImage(mapsource[value],2,2,96,46,x,y,100,50);
    });
  });
}
```

2D游戏基础-贴图

```
function renderMan() {
    context.drawImage(man,manleft,mantop,85,113.5,many,manx,85,113.5);
}
```

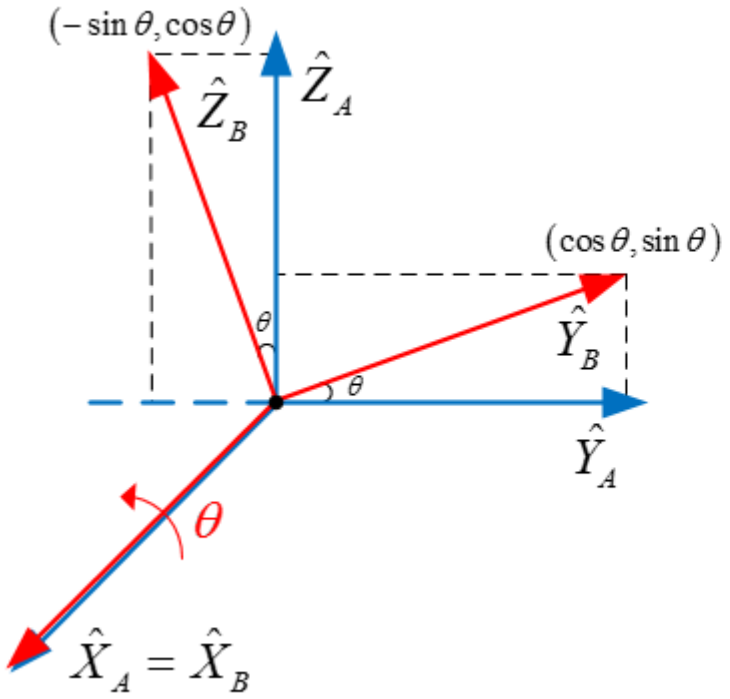
2D游戏基础-贴图

Canvas 帧动画

```
document.addEventListener('keydown',function(e){
 switch(e.which) {
   case 37:
     manx -= 5;
     many -= 10;
     mantop = 113.5 * 3;
     leftStatus = (leftStatus + 1) % 4;
     manleft = leftStatus * 85;
     break;
         ……
 }
});
```

目录

3D基础-旋转矩阵(例子！)



$$R_{rotx}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

3D基础-几何解释~

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos(\alpha) - y\sin(\alpha) \\ x\sin(\alpha) + y\cos(\alpha) \\ z \\ 1 \end{pmatrix}$$

3D基础-几何解释~

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos(\alpha) - y\sin(\alpha) \\ x\sin(\alpha) + y\cos(\alpha) \\ z \\ 1 \end{pmatrix}$$

3D基础-几何解释~

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos(\alpha) - y\sin(\alpha) \\ x\sin(\alpha) + y\cos(\alpha) \\ z \\ 1 \end{pmatrix}$$

3D基础-几何解释~

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos(\alpha) - y\sin(\alpha) \\ x\sin(\alpha) + y\cos(\alpha) \\ z \\ 1 \end{pmatrix}$$

目录

- Canvas 2D
  - 上手Canvas 2D – Danmaku
  - 2D 贴图~
- Canvas 3D
  - 基础理论
  - Three.js – 3D 直播例子
- Live 2D – 其实是3D的2D

3D 基础-THREE.JS

- 底层： 上述三个公式~ +OPENGLES 2.0（WEBGL）
- 抽象的对象
  - Camara 照相机、除了它 都在变。可以认为所有物体围着他反方向转。
  - Scene 场景。空间内一坨物体的集合。
  - Light 光源 & 材质 Textures
    - I = 环境光($I_{ambient}$) + 漫反射光($I_{diffuse}$) + 镜面高光($I_{specular}$)
    - $I_{ambient} = A_{intensity} * A_{color}$;
      - ($A_{intensity}$表示环境光强度，A_{color}表示环境光颜色)
    - $I_{diffuse} = D_{intensity} * D_{color} * N.L$;
      - ($D_{intensity}$表示漫反射强度，D_{color}表示漫反射光颜色，N为该点的法向量，L为光源向量)
    - $I_{specular} = S_{intensity} * S_{color} * (R.V)^n$;
    - ($S_{intensity}$表示镜面光照强度，S_{color}表示镜面光颜色，R为光的反射向量，V为观察者向量)

3D 基 础 -THREE.JS

其他支持

Mash 物体有顶点 边 面组成、描述物体形状的 顶点、边 面的几何

Bone 骨骼、在人、动物建模时。抽象出来的树型变换单位。

Audio / Video 等

AR / VR / 3D 直播

- 球状场景（直播视频 球状渲染）
- 视差
- 3D（单一球状渲染）-> VR（两个camara，产生视差,变成立体）-> AR 场景内渲染3D 模型，并模拟真实光照。

目录

- Canvas 2D
  - 上手Canvas 2D – Danmaku
  - 2D 贴图~
- Canvas 3D
  - 基础理论
  - Three.js – 3D 直播例子
- Live 2D – 其实是3D的2D

LIVE2D VS 3渲2技术

- Live 2D
  - 平面化的Mash & 骨骼（有付费工具和JS库。且兼容网页端、安卓、IOS）
    - 制作工具导出各个部分的动作。
    - JS调控幅度
  - 正前方的平行光 固定值光源。没有漫发射和镜面反射
  - 例子： fe / Bilibili live
- 3渲2 (暂时没有网页端开源渲染引擎，确切的说没有成熟的开源引擎。)
  - 3D立体模型
  - 特殊光照模型
  - 例子：动画 高校星歌剧 舰队collection 等。

Q & A

作业

- 推倒 位移矩阵 缩放矩阵。
- 思考下绕某一向量旋转（骨骼系统基础）
- 写个小游戏 galgame / RPG 2D 2.5D均可
- Deadline：没有，爱做不做。

引用

- 素材资源
  - http://www.2gei.com/view/8608.html 2.5D贴图素材
  - http://www.2gei.com/view/132.html
- 例子
  - http://fe.panda.tv/demos/player/2dgame/
  - http://fe.panda.tv/demos/player/live2d/
  - http://fe.panda.tv/demos/player/3dplayer/