1：单例模式实现 //data类，director类实现，

2：静态方法实现 //精灵类

3：ES6面向对象，类本身是一个函数，方法和属性作为原型链的方法存在的。

4：ES5面向对象，ES6面向对象

ES5继承实现（寄生组合继承）//调用一个对象的方法，用call,apply来用另一个对象替换

*//寄生组合继承* **function** *Person*(name,age) {  
 **this**.**name** = name;  
 **this**.**age** = age;  
 }  
 *Person*.prototype.Say = **function** () {  
 ***console***.log(**this**.**name**,**this**.**age**);  
 }  
  
 **let** s = **new** *Person*(**"张"**,12)  
 **let** s2 = {  
 **name**:**"小名"**,  
 **age**:12  
 }  
 s.Say.call(s2)

5：游戏实现

1：封装，资源封装（dataStore）,资源的加载（FristLoad）,背景图的绘制，地面的绘制

2：重绘API requestAnimationFrame(()=>this.run());

//在游览器刷新之前进行刷新，性能比较高

3：小游戏实现原理：整张图重绘（本质就是不断的重绘刷新）

//了解一下export export default的区别一个引人加{}，export default则不需要

4：绘制：

（1）背景图绘制

（2）地板绘制 解决问题：1：绘制位置问题，不停移动位置

(3)铅笔绘制 解决问题：随机的高度，绘制的高度，2组铅笔同时的显示（X坐标要一致），铅笔要被地板盖住（canvas绘制图形的位置问题），铅笔要能重复绘制，

//随机高度问题：top值设置一个中间数值，最高不能有一半高度，

**const** minTop = ***window***.innerHeight/8;  
**const** maxTop = ***window***.innerHeight/2;  
**const** top = minTop + ***Math***.random() \* (maxTop - minTop);

//铅笔同时对于，上下就是X坐标相同，重复绘制也就是，在进行创建一组铅笔进入数组，在前面绘制的数组超过边界的时候把其销毁。

**const** pencils = **this**.**dataStore**.get(**'pencils'**);  
 *//判断数组是否越界,X横坐标，初始值在窗口的右侧* **if**((pencils[0].**x** + pencils[0].**width**) <=0&& pencils.length ===4 ){  
 ***console***.log(2222);  
 *//pencils.length==4 代表是两组坐标，移出* pencils.shift();  
 pencils.shift();*//推出数组的第一个元素，并且把数组的个数减一* }  
*//创建* **if**(pencils[0].**x** <=(***window***.innerWidth - pencils[0].**width**)/2 && pencils.length === 2 ){  
 **this**.createPencil();  
 }

内存释放功能：添加一个GameOver属性 //用来清除定时器。释放内存

(4)小鸟的三种状态的绘制，剪裁的宽高，加速度重力下落的效果（这个很关键），点击事件会上升的效果，小鸟和铅笔模型的建立过程以及碰撞检测。

const speed = 0.2; //模拟切换的速度

this.count = this.count + speed;

//模拟的速度

if(this.index >=2){

this.count = 0; //每次到第三个小鸟在重复绘制

}

this.index = Math.floor(this.count); //向下摄取，保证index能在数组中去到

const g = 0.98/2.4;//重力

const offsetUp = 30; //保障小鸟下落的时候，能够先向上偏移一点距离

const offsetY = (g\*this.time\*(this.time - offsetUp)) / 2;

//遍历改变偏移；

for(let i=0;i<3;i++){

this.birdsY[i] = this.y[i] + offsetY;

}

this.time++;

//绑定一个事件，点击屏幕的时候，小鸟会上升

**this**.**index** = ***Math***.floor(**this**.**count**);  
**const** g = 0.98/2.4;*//重力***const** offsetUp = 30;  
**const** offsetY = (g\***this**.**time**\*(**this**.**time** - offsetUp)) / 2;

加速度：本质上随着时间的增加，下降的速度回越来越快其实本质是上，Y偏移量越来越大，但是开始有上升-y,本质是**const** offsetUp = 30;下降的时候抬升，利用吃时候，time和使用开始的位置可以让小鸟上升。

birdsEvent(){

//处理小鸟点击的时候向上飞；

for(let i=0;i<=2;i++){

this.dataStore.get('birds').y[i] = this.dataStore.get('birds').birdsY[i]; //真正偏移的高度

console.log(this.dataStore.get('birds').y[i]); //开始的高度

}

this.dataStore.get('birds').time = 0;

}

小鸟碰撞检测：地板检测和铅笔碰撞检测。

地板检测：判断条件：小鸟偏移Y坐标+他自己的高度超过了地板的Y坐标游戏结束。 if(this.dataStore.get('birds').birdsY[0]+this.dataStore.get('birds').birdsHeight[0] >=this.dataStore.get('land').y) { this.isGameOver = true; }

铅笔碰撞检测：

//绘制出小鸟的边框模型：小鸟的上下左右的位置，不是CSS的定位的属性，是canvas，原点的相对位置

const birdsBorder = {

top: birds.y[0],

bottom: birds.birdsY[0] + birds.birdsHeight[0],

left: birds.birdsX[0],

right: birds.birdsX[0] + birds.birdsWidth[0]

};

//绘制铅笔的位置。

//铅笔是循环里面的数值，因为铅笔的模型很多，上下一对，一共2租，鸟只有一个

for (let i = 0; i < length; i++) {

const pencil = pencils[i];

const pencilBorder = {

top: pencil.y,

bottom: pencil.y + pencil.height,

left: pencil.x,

right: pencil.x + pencil.width

};

if (Director.isStrike(birdsBorder, pencilBorder)) {

console.log('撞到水管啦');

this.isGameOver = true;

return;

}

}

//判断条件，小鸟的头部撞到铅笔的头部，小鸟的地步撞到下铅笔的地步，小鸟的左部分撞到铅笔的右部分，小鸟的右部分撞到铅笔的左部分。

(5)计分分数类

1：什么时候加分，在越过铅笔的时候开始加分（不能发生碰撞的时候进行检测）

2：控制加分的速度，因为canvas刷新评率很快所以需要设计一个flag来标志能否加分。

//经过检测之后，如果小鸟的左边超过了铅笔的右边没有碰撞上就可以算加分成功，