```
var Comm = require("../Comm.js");
var LevelPrefab = require("../prefab/LevelPrefab.js");
var LevelCurPrefab = require("../prefab/LevelCurPrefab.js");
cc.Class({
extends: cc.Component,
properties: {
levelLayout:cc.Node,
levelPrefab:cc.Prefab,
levelCurPrefab:cc.Prefab,
totalScoreLabel:cc.Label,
scrollView:cc.ScrollView,
},
// use this for initialization
onLoad: function () {
// cocosAnalytics.enableDebug(true);
// console.log("登录统计");
// cocosAnalytics.CAAccount.loginStart();
// cocosAnalytics.CAAccount.loginSuccess({'userID':'dddddddd'});
// 读取分数表
Comm.readLevelScores();
Comm.calcScoreLogic();
this.totalScoreLabel.string = "总分:" + Comm.totalScore;
for (var i = 1; i < Comm.minScoreLevel; i++) {
var cell = cc.instantiate(this.levelPrefab);
cell.getComponent(LevelPrefab).setLevel(i);
cell.getComponent(LevelPrefab).setClickCallback(function(level){
// 开始玩这一关
Comm.currentLevel = level;
console.log("Comm.currentLevel", Comm.currentLevel);
cc.director.loadScene("GameScene");
});
this.levelLayout.addChild(cell);
}
// 推荐关卡
var cell = cc.instantiate(this.levelCurPrefab);
cell.getComponent(LevelCurPrefab).setLevel(Comm.minScoreLevel);
cell.getComponent(LevelCurPrefab).setClickCallback(function(level){
// 开始玩这一关
Comm.currentLevel = Comm.minScoreLevel;
console.log("Comm.currentLevel", Comm.currentLevel);
cc.director.loadScene("GameScene");
});
this.levelLayout.addChild(cell);
for (var i = Comm.minScoreLevel+1; i <= Comm.maxLevel; i++) {
var cell = cc.instantiate(this.levelPrefab);
cell.getComponent(LevelPrefab).setLevel(i);
cell.getComponent(LevelPrefab).setClickCallback(function(level){
// 开始玩这一关
Comm.currentLevel = level;
console.log("Comm.currentLevel", Comm.currentLevel);
```

```
cc.director.loadScene("GameScene");
});
this.levelLayout.addChild(cell);
}
// 未解锁
var cell = cc.instantiate(this.levelPrefab);
cell.getComponent(LevelPrefab).setUnlockInfoForLevel(Comm.maxLevel + 1);
cell.getComponent(LevelPrefab).setClickCallback(function(level){
Comm.tip("总分达到"+ Comm.calcTargetScore(level-1) + "才能玩这一关");
});
this.levelLayout.addChild(cell);
this.scheduleOnce(function() {
var maxScrollOffset = this.scrollView.getMaxScrollOffset();
var percent = Math.min(1, Math.max(0,(Comm.minScoreLevel - 6) /
(Comm.maxLevel+1 - 7)));
console.log("percent",percent);
this.scrollView.scrollToOffset(cc.p(0,maxScrollOffset.y*percent), 0.3);
}, 0);
console.log("弄一下统计的东西");
console.log("typeof", typeof(anysdk));
if (typeof(anysdk) != "undefined") {
var agent = anysdk.agentManager;
console.log("agent",agent);
if (agent) {
var user_plugin = agent.getAnalyticsPlugin();
console.log("user_plugin",user_plugin);
if (user_plugin) {
if (user_plugin.setAccount) {
user_plugin.setAccount({
Account_Id : "123456",
Account_Name : "test",
Account_Type : (anysdk.AccountType.ANONYMOUS).toString(),
Account_Level : "0",
Account_Age : "0",
Account_Operate : (anysdk.AccountOperate.LOGIN).toString(),
Account_Gender : (anysdk.AccountGender.UNKNOWN).toString(),
Server_Id : "0"
});
}
if (user_plugin.startSession) {
user_plugin.startSession();
}
console.log("统计的东西弄完了")
}
}
}
},
startButtonClick: function(){
},
});
```

```
var TipPrefab = require("./prefab/TipPrefab.js");
var ConfirmDialogPrefab = require("./prefab/ConfirmDialogPrefab.js");
module.exports={
// 网格大小
gridSize:10,
// 计算消除得分
calcClearScore: function (count){
return count * count * 5;
},
// 计算剩余水果所能得的分
calcLastScore: function(count){
return Math.max(0, 2000 - count*count*20)
},
// 显示一个 tip
tip: function(tipStr){
console.log("tip")
// 加载 Prefab
cc.loader.loadRes("prefab/TipPrefab", function (err, prefab) {
if (err) {
console.log(err);
return;
}
var newNode = cc.instantiate(prefab);
cc.director.getScene().addChild(newNode);
newNode.getComponent(TipPrefab).show(tipStr);
});
},
// 弹出一个确认框
confirm: function(title, content, btn1Str, btn1Cb, btn2Str, btn2Cb){
console.log("confirm")
// 加载 Prefab
cc.loader.loadRes("prefab/ConfirmDialogPrefab", function (err, prefab) {
if (err) {
console.log(err);
return;
}
var newNode = cc.instantiate(prefab);
cc.director.getScene().addChild(newNode);
newNode.getComponent(ConfirmDialogPrefab).show(title, content, btn1Str,
btn1Cb, btn2Str, btn2Cb);
});
},
// 读取所有关卡的分数
readLevelScores:function() {
var strScores = cc.sys.localStorage.getItem("LEVEL_SCORES");
if (strScores) {
this.levelScores = JSON.parse(strScores);
}else{
this.levelScores = {};
}
```

```
// cc.sys.localStorage.setItem("LEVEL_SCORES", null);
},
// 保存所有关卡的分数
saveLevelScores:function() {
var strScores = JSON.stringify(this.levelScores);
cc.sys.localStorage.setItem("LEVEL_SCORES", strScores);
},
// 计算分数逻辑，总分，最低分关卡，下一关解锁信息等
calcScoreLogic:function() {
// 总分
this.totalScore = 0;
// 最低分关卡
this.minScoreLevel = 0;
// 倒数第二分关卡
this.min2ScoreLevel = 0;
// 总分
for(var i in this.levelScores) {
var score = this.levelScores[i];
this.totalScore += score; // 总分
}
// 最大关卡
var maxLevel = 0;
console.log(this.totalScore, this.calcTargetScore(maxLevel));
while(this.totalScore >= this.calcTargetScore(maxLevel)) {
maxLevel++;
}
this.maxLevel = maxLevel;
// 最低分关卡的得分
var minScore = Number.POSITIVE_INFINITY;
// 倒数第二分关卡的得分
var min2Score = Number.POSITIVE_INFINITY;
for(var i = 1; i <= this.maxLevel; i++){
var score = this.levelScores[i.toString()];
if (!score){score=0;}
if (score < minScore || (score == minScore && this.minScoreLevel > i)) {
minScore = score;
this.min2ScoreLevel = this.minScoreLevel;
this.minScoreLevel = i;
} else if (score < min2Score || (score == min2Score && this.min2ScoreLevel > i)) {
min2Score = score;
this.min2ScoreLevel = i;
}
}
},
// 计算关卡目标分数
calcTargetScore: function(level) {
if (!this.levelCfg){this.levelCfg = {}}
if (this.levelCfg[level.toString()]){return this.levelCfg[level.toString()]}
var r = 1000;
if (level > 1) {
```

```
r = this.calcTargetScore(level-1);
r = r + (0 == r % 2 ? 2000 : 3000);
r = r + Math.floor(level / 10) * 300;
}else if(level == 0){
r = 0;
}
this.levelCfg[level.toString()] = r;
return r;
},
// 设置某一关的分数
setLevelScore: function(level, score) {
console.log("setLevelScore", level, score);
this.levelScores[level.toString()] = score;
},
// 计算目标分的字符串
calcTargetStr: function() {
this.targetStrTab = {};
// 超过倒数第二关的目标分
var target1Score = 0;
if (this.min2ScoreLevel && this.min2ScoreLevel != 0){
target1Score = this.levelScores[this.min2ScoreLevel];
if (target1Score) {
target1Score ++; // 要超过倒数第二，所以得加一分，要不然不能算超过
} else {
target1Score = 0;
}
}
var target2Score = 0;
if (this.maxLevel){
// 目标分=解锁分-(总分-当前关分)
var curScore = this.levelScores[this.currentLevel];
if (! curScore) {
curScore = 0;
}
target2Score = this.calcTargetScore(this.maxLevel)-(this.totalScore-curScore);
}
if (target1Score > target2Score) {
if (target2Score > 0){
this.targetStrTab.littleTarget = target2Score;
this.targetStrTab.littleTargetStr = "(解锁第" + (this.maxLevel+1) + "关)\n";
this.targetStrTab.bigTarget = target1Score;
this.targetStrTab.bigTargetStr = "(超过第" + this.min2ScoreLevel + "关)";
} else {
this.targetStrTab.oneTarget = target1Score;
this.targetStrTab.oneTargetStr = "(超过第" + this.min2ScoreLevel + "关)";
}
} else {
if (target1Score > 0){
this.targetStrTab.littleTarget = target1Score;
this.targetStrTab.littleTargetStr = "(超过第" + this.min2ScoreLevel + "关)";
```

```
this.targetStrTab.bigTarget = target2Score;
this.targetStrTab.bigTargetStr = "(解锁第" + (this.maxLevel+1) + "关)\n";
} else {
this.targetStrTab.oneTarget = target2Score;
this.targetStrTab.oneTargetStr = "(解锁第" + (this.maxLevel+1) + "关)\n";
}
}
},
};
var StarPrefab = require("../prefab/StarPrefab.js");
var Comm = require("../Comm.js");
cc.Class({
extends: cc.Component,
properties: {
starPrefab:cc.Prefab,
starGrid:cc.Node,
scoreLabel:cc.Label,
scorePreLabel:cc.Label,
targetLabel:cc.Label,
},
// use this for initialization
onLoad: function () {
// cocosAnalytics.enableDebug(true);
console.log("登录统计");
// cocosAnalytics.CAAccount.loginStart();
// cocosAnalytics.CAAccount.loginSuccess({'userID':'dddddddd'});
var self = this;
this.starGame();
this.starGrid.on(cc.Node.EventType.TOUCH_START, function(e){
var pos = self.starGrid.convertToNodeSpace(e.touch.getLocation());
self.touchStar(parseInt(pos.x / (self.starGrid.width/10)),parseInt(pos.y /
(self.starGrid.height/10)));
});
// 适配屏幕
var size = cc.director.getWinSize();
this.scoreLabel.node.y = size.height/2 - (size.height-size.width)/2/2;
this.targetLabel.node.y = size.height/2 - 50;
},
// 开始游戏
starGame: function () {
this.initStatus();
this.initGrid();
Comm.calcTargetStr()
this.targetButtonClick();
this.updateTargetLabel();
},
// 初始化状态变量
initStatus: function () {
// 存放状态变量的对象
this.stVar = {};
```

```
// 是否在选中状态
this.stVar.selected = false;
// 总分
this.totalScore = 0;
},
// 生成网格
initGrid: function () {
// 清空原来的
if (this.gridStarUi) {
for (var x = 0; x < Comm.gridSize; x++) {
for (var y = 0; y < Comm.gridSize; y++) {
if (this.gridStarUi[x][y]) {
this.gridStarUi[x][y].destroy();
}
}
}
}
// 网格数据
var gridData = [];
// 网格水果 ui
var gridStarUi = [];
for (var i = 0; i < Comm.gridSize; i++) {
gridData[i] = [];
gridStarUi[i] = [];
for (var j = 0; j < Comm.gridSize; j++) {
var star = cc.instantiate(this.starPrefab);
var starClass = star.getComponent(StarPrefab);
star.parent = this.starGrid;
starClass.setGridXY(i,j);
var rnd = parseInt(Math.random()*5 + 1);
starClass.setType(rnd);
gridData[i][j] = rnd;
gridStarUi[i][j] = star;
}
}
this.gridData = gridData;
this.gridStarUi = gridStarUi;
},
// 更新目标分 label
updateTargetLabel: function () {
if (Comm.targetStrTab.oneTarget) {
if (this.totalScore >= Comm.targetStrTab.oneTarget) {
this.targetLabel.string = "目标:" + Comm.targetStrTab.oneTarget + "分(已完成)";
} else {
this.targetLabel.string = "目标:" + Comm.targetStrTab.oneTarget + "分";
}
}else{
if (this.totalScore < Comm.targetStrTab.littleTarget) {
this.targetLabel.string = "小目标:" + Comm.targetStrTab.littleTarget + "分";
} else if(this.totalScore < Comm.targetStrTab.bigTarget){
```

```
this.targetLabel.string = "大目标:" + Comm.targetStrTab.bigTarget + "分";
} else {
this.targetLabel.string = "大目标:" + Comm.targetStrTab.bigTarget + "分(已完成)";
}
}
},
// 点击了一个水果，xy 为数据坐标
touchStar: function (x,y) {
if (this.stVar.selected) {
if (this.connectContain(x,y)) {
// 如果点击了已被选中的水果
this.cleanOnce(x, y);
this.stVar.selected = false;
} else {
// 如果点击了未被选中的水果
this.setConnectStarSelect(false);
this.scorePreLabel.node.active = false;
this.touchStar(x,y);
}
} else {
if (this.gridData[x][y] == 0) {return;}
// 相连的水果
this.stVar.connectStars = [[x, y]];
this.checkStar(x, y);
if (this.stVar.connectStars.length >= 2) {
this.setConnectStarSelect(true);
this.scorePreLabel.string = Comm.calcClearScore(this.stVar.connectStars.length);
this.scorePreLabel.node.stopAllActions();
this.scorePreLabel.node.setPosition(this.gridStarUi[x][y].position);
this.scorePreLabel.node.opacity = 255;
this.scorePreLabel.fontSize = 32;
this.scorePreLabel.node.active = true;
}
}
},
// 递归查找相连的水果
checkStar: function (x, y) {
var starType = this.gridData[x][y];
// 要扫描的 4 个水果（上下左右）
var scanStar = [[x+1, y],[x-1, y],[x, y-1],[x, y+1]];
for (var i = scanStar.length - 1; i >= 0; i--) {
scanStar[i]
var scanX = scanStar[i][0];
var scanY = scanStar[i][1];
if (this.inGrid(scanX, scanY)
&& this.gridData[scanX][scanY] == starType
&& (! this.connectContain(scanX, scanY))
) {
this.stVar.connectStars[this.stVar.connectStars.length] = [scanX, scanY];
this.checkStar(scanX, scanY);
```

```
}
}
},
// 是否在网格范围内
inGrid: function (x,y) {
return x >= 0 && x < Comm.gridSize && y >= 0 && y < Comm.gridSize
},
// 相连数组里是否有该坐标
connectContain: function(x, y){
for (var i = this.stVar.connectStars.length - 1; i >= 0; i--) {
var star = this.stVar.connectStars[i];
if (star[0] == x && star[1] == y) {
return true;
}
}
return false;
},
// 选中或取消选中，相连的水果
setConnectStarSelect: function (selected) {
for (var i = this.stVar.connectStars.length - 1; i >= 0; i--) {
var star = this.stVar.connectStars[i];
this.gridStarUi[star[0]][star[1]].getComponent(StarPrefab).setSelected(selected);
}
this.stVar.selected = selected;
},
// 做一次消除操作
cleanOnce: function () {
var self = this;
// 计算得分
this.scorePreLabel.fontSize = 64;
this.scorePreLabel.node.runAction(cc.sequence(
cc.moveBy(0.2, cc.p(0, 20)),
cc.delayTime(0.8),
cc.spawn(cc.moveBy(0.2,cc.p(0, 10)), cc.fadeOut(0.2))
));
this.totalScore += parseInt(this.scorePreLabel.string);
this.scoreLabel.string = this.totalScore;
this.updateTargetLabel();
// 清除水果
this.clearConnectStar();
// 让水果下落
this.fallDownStar();
// 让水果往左靠
this.fallLeftStar();
// 检测本关是否结束
if (this.checkOver()){
console.log("不能再消除了");
var lastCount = this.checkCount();
var lastCountScore = Comm.calcLastScore(lastCount);
this.totalScore += lastCountScore;
```

```
this.scoreLabel.string = this.totalScore;
console.log("计算剩余水果分");
console.log("最终得分", this.totalScore);
// 目标完成情况
var dlgStr = "";
if (Comm.targetStrTab.oneTarget) {
var isOk = "已完成";
if (this.totalScore < Comm.targetStrTab.oneTarget) {isOk="未完成";}
dlgStr = "目标:" + Comm.targetStrTab.oneTarget + "分" +
Comm.targetStrTab.oneTargetStr + isOk;
} else {
var isOk1 = "已完成";
if (this.totalScore < Comm.targetStrTab.littleTarget) {isOk1="未完成";}
var isOk2 = "已完成";
if (this.totalScore < Comm.targetStrTab.bigTarget) {isOk2="未完成";}
dlgStr = "小目标:" + Comm.targetStrTab.littleTarget + "分" +
Comm.targetStrTab.littleTargetStr + isOk1 +"\n"
+ "大目标:" + Comm.targetStrTab.bigTarget + "分" +
Comm.targetStrTab.bigTargetStr + isOk2;
}
// 历史最高分
var lastScore = Comm.levelScores[Comm.currentLevel.toString()]
if (!lastScore || this.totalScore > lastScore) {
console.log("破记录了");
Comm.setLevelScore(Comm.currentLevel, this.totalScore);
// Comm.levelScores[Comm.currentLevel.toString()] = this.totalScore;
Comm.saveLevelScores();
Comm.calcScoreLogic();
}
Comm.confirm(
"不能再消除了",
"您的得分:" + this.totalScore + "(其中剩余水果"+ lastCount + "附加分:" +
lastCountScore + ")",
"确定",function(){
Comm.confirm(
"目标",
dlgStr,
"回主菜单",function(){
cc.director.loadScene("MainScene");
});
});
}
},
// 清除相连数组里的水果
clearConnectStar: function () {
for (var i = this.stVar.connectStars.length - 1; i >= 0; i--) {
var star = this.stVar.connectStars[i];
this.gridData[star[0]][star[1]] = 0;
this.gridStarUi[star[0]][star[1]].destroy();
}
```

```
},
// 让水果下落
fallDownStar: function () {
// 遍历每一列
for (var x=0; x < Comm.gridSize; x++) {
// 下落的距离
var fallDistance = 0;
// 从下往上，遍历每一个水果
for (var y = 0; y < Comm.gridSize; y++){
// 如果是空，则增加一个下落距离
if (this.gridData[x][y] == 0){
fallDistance++;
}
// 如果需要下落且当前不是空，就记录到下落数组里
if (fallDistance > 0 && this.gridData[x][y] > 0){
this.gridData[x][y - fallDistance] = this.gridData[x][y];
this.gridData[x][y] = 0;
this.gridStarUi[x][y].getComponent(StarPrefab).goTo(x,y - fallDistance, 0);
this.gridStarUi[x][y - fallDistance] = this.gridStarUi[x][y];
this.gridStarUi[x][y] = null;
}
}
}
},
// 让水果左靠
fallLeftStar: function () {
// 左靠距离
var fallLeftDistance = 0;
for (var x = 0; x < Comm.gridSize; x++) {
// 如果最底下的水果是空，则整列都是空，就加一个距离
if (this.gridData[x][0] == 0) {
fallLeftDistance++;
}
// 如果该列有水果，并且需要左靠
if (this.gridData[x][0] != 0 && fallLeftDistance != 0) {
// 执行左靠
for (var y = 0; y < Comm.gridSize; y++) {
if (this.gridData[x][y] > 0) {
this.gridData[x - fallLeftDistance][y] = this.gridData[x][y];
this.gridData[x][y] = 0;
this.gridStarUi[x][y].getComponent(StarPrefab).goTo(x - fallLeftDistance, y, 0.1);
this.gridStarUi[x - fallLeftDistance][y] = this.gridStarUi[x][y];
this.gridStarUi[x][y] = null;
} else {
break;
}
}
}
}
},
```

```
checkOver: function() {
// 先遍历列，效率高一些，因为如果一个水果为空，那上面一定没有水果了
for (var x=0; x < Comm.gridSize; x++){
for (var y=0; y < Comm.gridSize; y++){
var starType = this.gridData[x][y];
if (starType == 0){
break;
}
// 要扫描的 4 个水果（上下左右）
var scanStar = [[x+1, y],[x-1, y],[x, y-1],[x, y+1]];
for (var i = 0; i < scanStar.length; i++) {
// 如果被扫描的 4 个中有相连的，就直接返回 false
var tmpX = scanStar[i][0];
var tmpY = scanStar[i][1];
if (this.inGrid(tmpX, tmpY) && this.gridData[tmpX][tmpY] == starType){
return false
}
}
}
}
return true
},
// 检测剩余多少个水果
checkCount: function() {
var count = 0;
for (var x=0; x < Comm.gridSize; x++){
for (var y=0; y < Comm.gridSize; y++){
var starType = this.gridData[x][y];
if (starType > 0){
count ++;
}
}
}
return count;
},
// 点击目标分
targetButtonClick:function () {
var dlgStr = "";
if (Comm.targetStrTab.oneTarget) {
dlgStr = "目标:" + Comm.targetStrTab.oneTarget + "分" +
Comm.targetStrTab.oneTargetStr;
} else {
dlgStr = "小目标:" + Comm.targetStrTab.littleTarget + "分" +
Comm.targetStrTab.littleTargetStr + "\n"
+ "大目标:" + Comm.targetStrTab.bigTarget + "分" +
Comm.targetStrTab.bigTargetStr;
}
console.log(dlgStr);
Comm.confirm(
"目标",
```

```
dlgStr,
"确定",
function(){},
);
}
});
cc.Class({
extends: cc.Component,
properties: {
titleLabel:cc.Label,
contentLabel:cc.Label,
btn1Node:cc.Node,
btn2Node:cc.Node,
btn1Label:cc.Label,
btn2Label:cc.Label,
},
// 弹出对话框
show: function (title, content, btn1Str, btn1Cb, btn2Str, btn2Cb) {
this.titleLabel.string = title;
this.contentLabel.string = content;
this.btn1Label.string = btn1Str;
this.btn1Cb = btn1Cb;
if(!btn2Str) {
this.btn1Node.x = 0;
this.btn2Node.active = false;
}else{
this.btn2Label.string = btn2Str;
this.btn2Cb = btn2Cb;
}
},
btn1Click:function() {
this.btn1Cb();
this.node.destroy();
},
btn2Click:function() {
this.btn2Cb();
this.node.destroy();
},
// 点击空白
fullScreenBtnClick:function() {
console.log("fullScreenBtnClick");
},
// 点击对话框内
dialogBtnClick:function() {
console.log("dialogBtnClick");
},
});
var Comm = require("../Comm.js");
cc.Class({
extends: cc.Component,
```

```
properties: {
strLabel:cc.Label,
},
setLevel:function (level) {
this.level = level;
var score = Comm.levelScores[level.toString()];
if (!score){score=0;}
this.strLabel.string = "第"+level+"关 "+score+"分";
},
mainButtonClick: function() {
this.cb(this.level);
},
setClickCallback: function(cb) {
this.cb = cb;
},
});
var Comm = require("../Comm.js");
cc.Class({
extends: cc.Component,
properties: {
strLabel:cc.Label,
lock:cc.Node,
},
setLevel:function (level) {
this.level = level;
var score = Comm.levelScores[level.toString()];
if (!score){score=0;}
this.strLabel.string = "第"+level+"关 "+score+"分";
this.lock.active = false;
},
setUnlockInfoForLevel:function (level) {
this.level = level;
this.strLabel.string = "总分"+Comm.calcTargetScore(this.level-1)+"解锁第" +
level + "关";
this.lock.active = true;
},
mainButtonClick: function() {
this.cb(this.level);
},
setClickCallback: function(cb) {
this.cb = cb;
},
});
cc.Class({
extends: cc.Component,
properties: {
pic1:cc.SpriteFrame,
pic2:cc.SpriteFrame,
pic3:cc.SpriteFrame,
pic4:cc.SpriteFrame,
```

```
pic5:cc.SpriteFrame,
},
// use this for initialization
onLoad: function () {
var self = this;
this.node.setScale(1.1);
// this.node.on(cc.Node.EventType.TOUCH_START, function(e){
//         self.ctrl.touchStar(self.gridX, self.gridY);
// });
},
//  设置方块类型
setType: function (starType) {
this.starType = starType;
this.getComponent(cc.Sprite).spriteFrame = this["pic" + starType];
},
//  设置网格坐标
setGridXY: function (x,y) {
this.gridX = x;
this.gridY = y;
this.node.setPosition((x-5)*75 + 75/2, (y-5)*75 + 75/2);
},
setSelected:function(selected) {
if (!selected) {
this.node.stopAllActions();
this.node.setScale(1.1);
} else {
this.node.runAction(
cc.repeatForever(cc.sequence(
cc.scaleTo(0.3, 1.1),
cc.scaleTo(0.3, 0.95)
)));
}
},
//  移动到一个网格坐标
goTo: function (x,y, delay) {
this.gridX = x;
this.gridY = y;
this.node.runAction(cc.sequence(
cc.delayTime(delay),
cc.moveTo(0.1, cc.p((x-5)*75 + 75/2, (y-5)*75 + 75/2))
));
}
});
cc.Class({
extends: cc.Component,
properties: {
tipLabel:cc.Label,
tipBg:cc.Node,
},
//  显示一个 tip
```

```
show: function (tipStr) {
var self = this;
this.tipLabel.string = tipStr;
this.node.setPosition(cc.director.getWinSize().width/2, 100);
this.node.runAction(cc.sequence(
cc.delayTime(2),
cc.fadeOut(0.3),
cc.callFunc(function(){
self.node.destroy();
})
));
},
});
```