用 Cocos Creator 快速制作打地鼠游戏

前言

在开始构建我们的游戏之前，让我们从GitHub下载教程。你也可以下载已完成的版本，但是首先尝试与我们一起构建你的游戏。

如果你在我们的教程中遇到麻烦，请学习一下文档：

- 准备开发环境
  - Cocos Creator 2.1.2
  - Visual Studio Code
- 文档
  - Animation system
  - Collision system
- API
  - Animation
  - CollisionManager

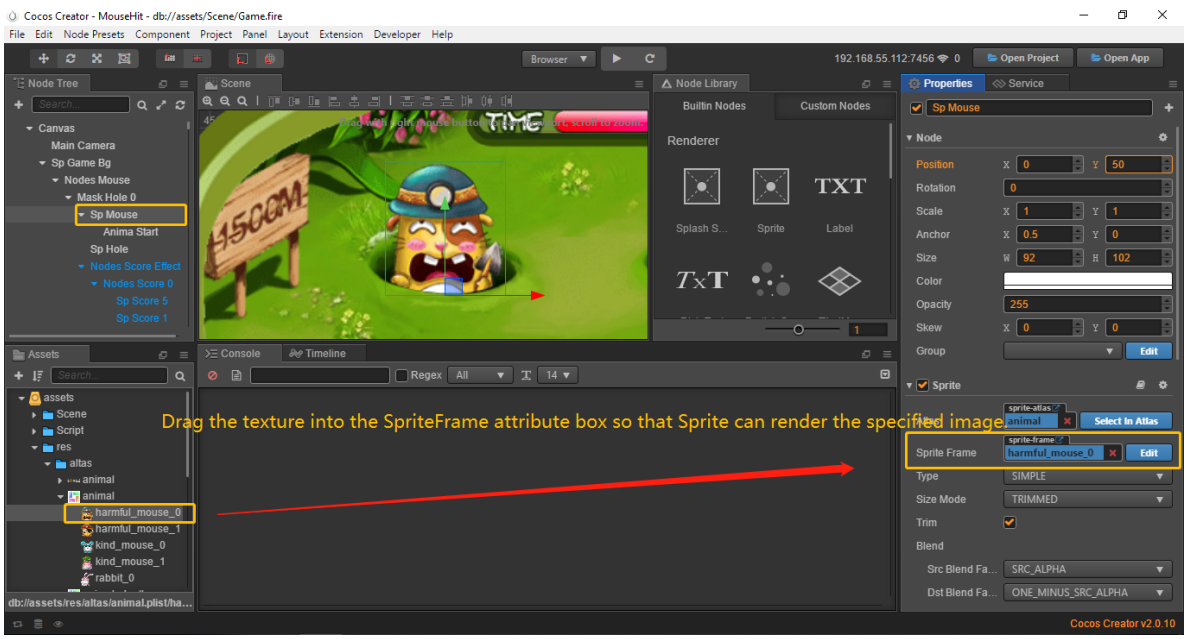
我们开始吧！

1. 使用动画系统

我们可以使用 CocosCreator 中的动画编辑器制作一个简单的老鼠进出的动画。

1.1 添加老鼠精灵到场景中:

我们需要做的第一件事就是把 Sprite 添加到游戏中。为此，我们需要在 Sp Mouse 中创建一个节点，该节点还不能显示图像。让我们在 Sp Mouse 中添加一个 Sprite 组件，它可以将指定的图片渲染到屏幕上。然后，我们所要做的就是将图片 “harmful_mouse_0” 拖到 Sprite 组件中的 SpriteFrame 属性框中。

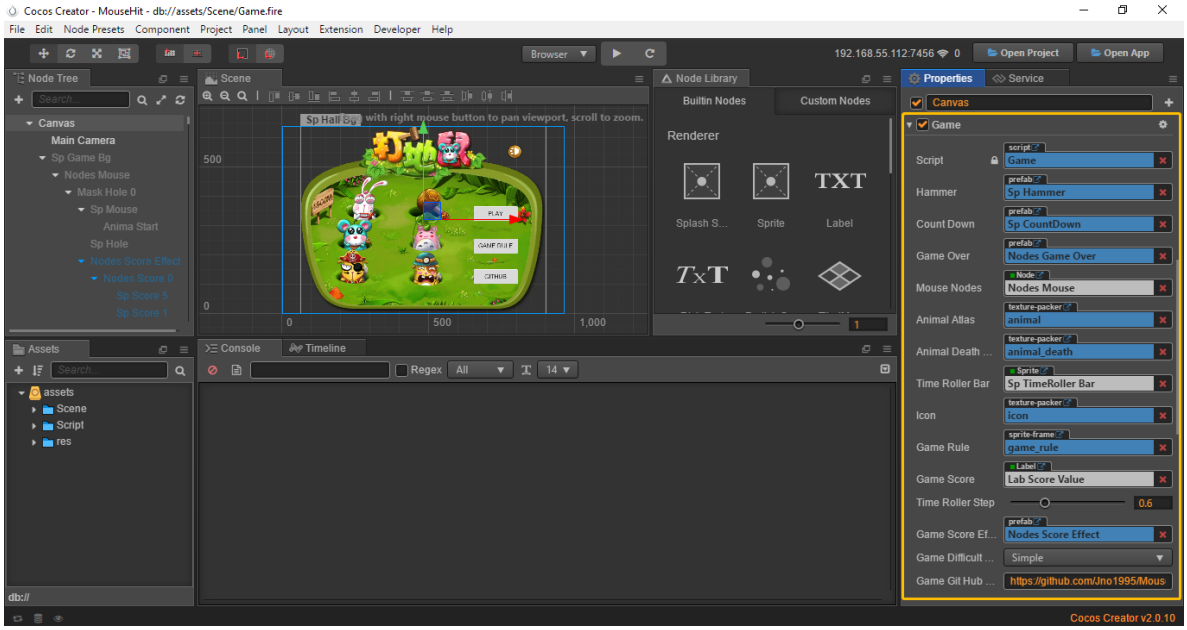


1.1.1 添加游戏逻辑：

现在，我们需要将游戏脚本game.js添加到游戏中，并向其添加所需的属性检查器参数：

1.1.2 Game.js 的代码:

请将以下代码添加到Game.js中



```
var Difficult = cc.Enum({
    Simple: 1000,
    Ordinary: 2500,
    Difficult: 5000
});

properties: {
        hammer: {
            default: null,
            type: cc.Prefab
        },

        countDown: {
            default: null,
```

```
        type: cc.Prefab
    },

    gameOver: {
        default: null,
        type: cc.Prefab
    },

    mouseNodes: {
        default: null,
        type: cc.Node
    },

    animalAtlas: {
        default: null,
        type: cc.SpriteAtlas
    },

    animalDeathAtlas: {
        default: null,
        type: cc.SpriteAtlas
    },

    timeRollerBar: {
        default: null,
        type: cc.Sprite
    },

    icon: {
        default: null,
        type: cc.SpriteAtlas
    },

    gameRule: {
        default: null,
        type: cc.SpriteFrame
    },

    gameScore: {
        default: null,
        type: cc.Label
    },

    timeRollerStep: {
        default: 1,
        range: [0, 2, 0.1],
        slide: true
    },

    gameScoreEffect: {
        default: null,
        type: cc.Prefab
    },

    gameDifficultScore:{
        default: Difficult.Simple,
        type: Difficult,
        tooltip: "Simple:2000\n Ordinary:4000\n Difficult:6000",
```

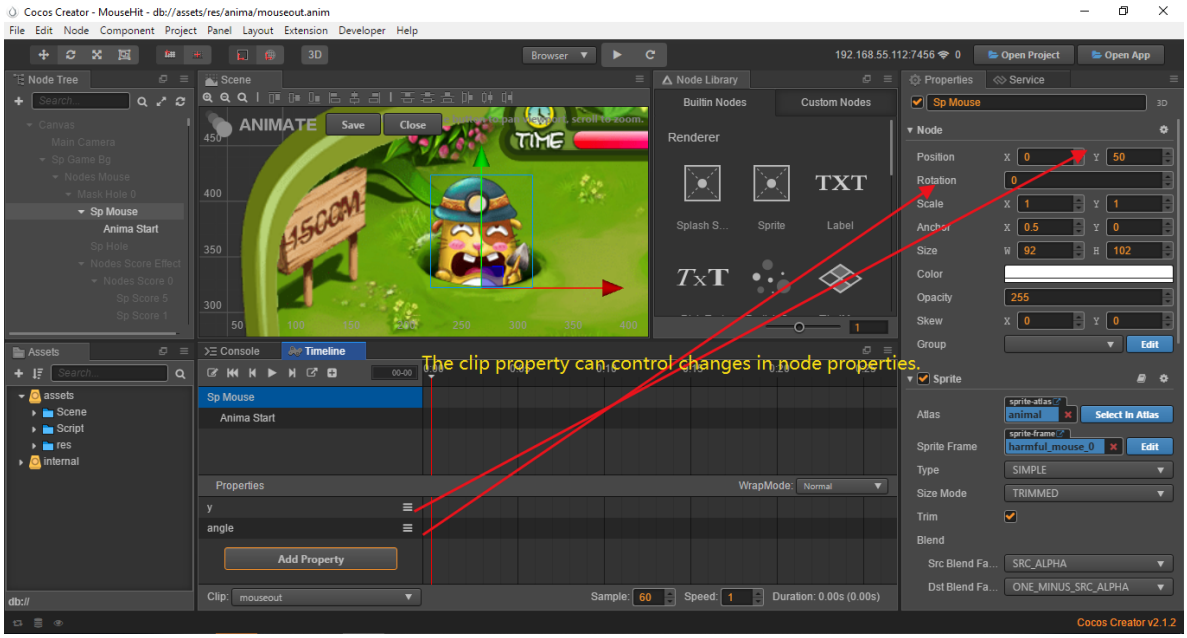
```
        },

        gameGitHubUrl:"",

        _isCollider: false,
        _mouseNode: null,
        _mouseIndexArr: [],
        _score: 0
},
```

1.2 添加动画效果到老鼠上：

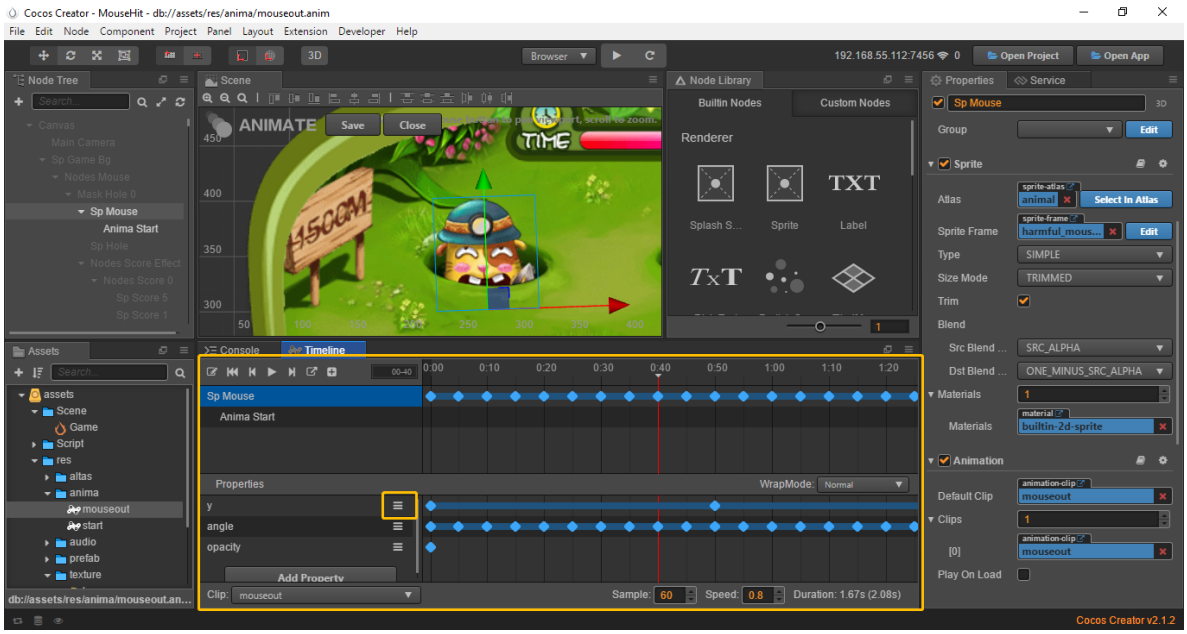
现在，我们需要向节点添加动画组件，并在资源文件夹中创建动画资源命名为mouseout。当节点属性更改时，我们刚刚创建了记录，动画组件可以读取这个数据使节点发生变化。在资源部分打开anima文件夹并选择 ”mouseout“ 动画帧。编辑器默认是不打开动画编辑器的，我们点击动画编辑按钮进入动画编辑器并且给 mouseout 动画添加两个属性: Y 和 angle 。



1.3 给 Clip 插入关键帧并调整动画效果

Clip 属性的右侧有一个小按钮，点击 ’insert key frame’ 之后，可以在指定的时间轴上创建动画帧。当它被创建时，与 Clip 帧属性相关的节点属性会被记录在 Clip 中。如果我们继续在后面的时间轴中添加具有不同 Clip 属性的动画帧，则可以生成连续的动画变化。

此时，我们希望老鼠不断地从鼠洞中晃动出现、消失，所以我们可以在时间轴为 `00:00`、`00:50`、`01:40` 的位置，将相应的 Y 坐标设置为 `-50`、`0`、`50` 。因为我们想要鼠标不断地抖动，所以需要在多个时间段内设置不同的角度，相应的angle等于 `-2`、`0`、`2`。



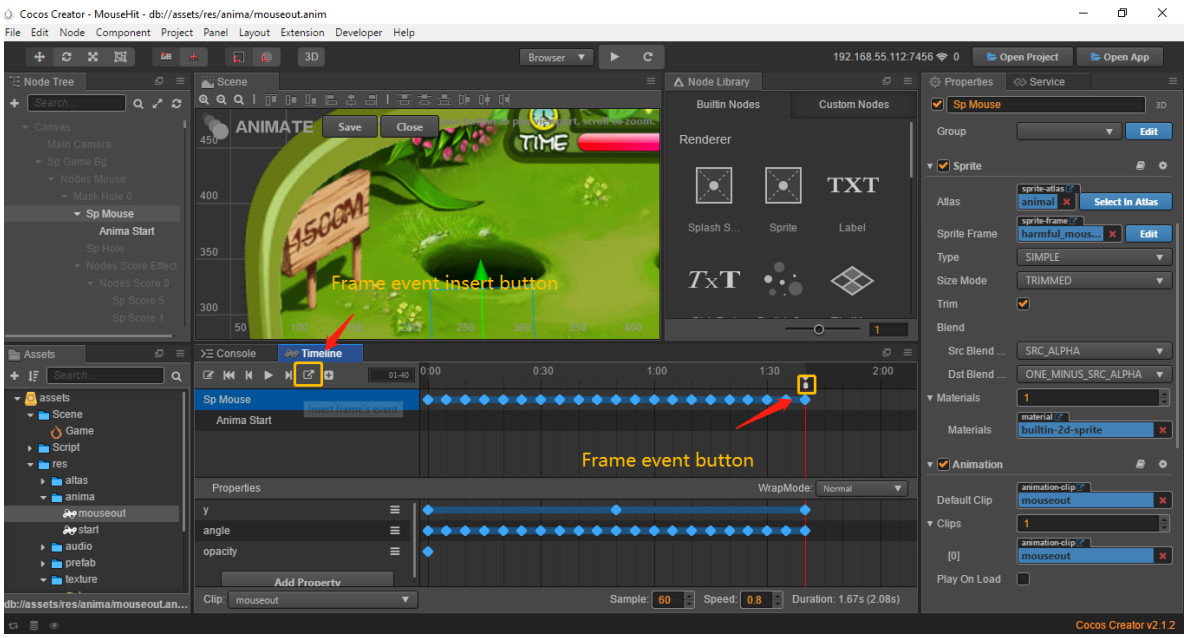
1.4 继续为这个动画帧添加帧回调

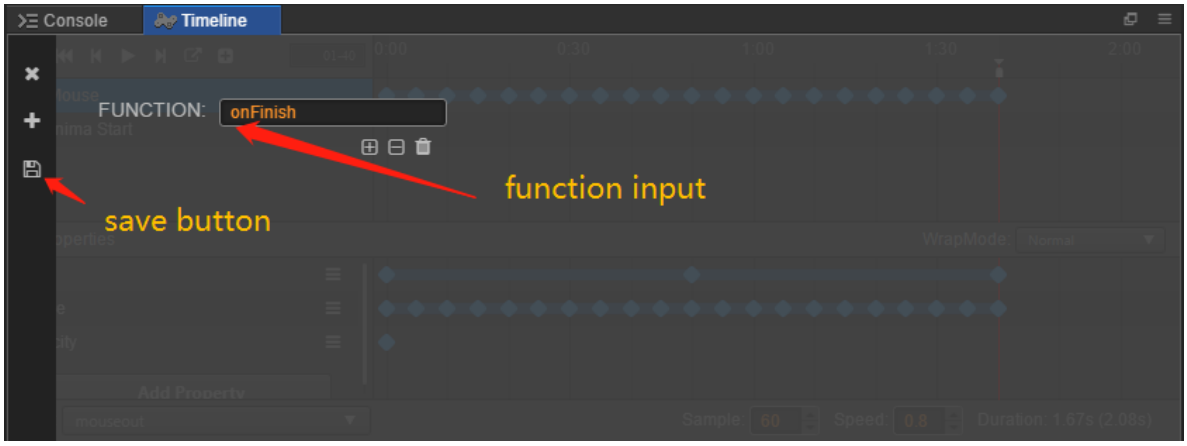
我们首先创建一个名为 AnimationCallBack.js 的 JS 脚本，在 cc.Class 中添加一个脚本命名为 'onFinish'，并将其添加到 Sp Mouse 节点上：

1.4.1 代码：

```
onFinish () {
    this.node.opacity = 0;
    this.node.getChildByName("Anima Start").opacity = 0;
    this.node.getChildByName("Anima Start").getComponent(cc.Animation).stop();
    this.node.getComponent(cc.BoxCollider).enabled = false;
},
```

回到动画编辑器中，我们选择最后一帧的时间轴，然后点击插入帧事件，以便将帧事件插入到当前帧中。右击生成的帧事件图标，选择 'Edit'，进入帧事件编辑界面。在输入框中输入 'onFinish'，然后单击'保存'按钮退出动画编辑器模式，这样就成功添加帧事件了。





此时，我们已经完成了老鼠精灵的动画设置。

---

2 把鼠标变成锤子

老鼠需要用锤子来击晕，所以我们把鼠标变成锤子，这样鼠标在任何地方移动，都可以随时击晕老鼠。

2.1 通过监听系统事件将鼠标变成锤子。

系统事件是最基本的游戏输入方式，每个游戏都应该很好地管理自己的系统事件。Cocos Creator 引擎封装了系统事件的管理，我们使用on函数来实现监听所有系统事件。当监听用户触摸事件或鼠标事件时，我们可以通过访问回调参数`event`来访问当前游戏所接收到的触摸对象数据。这是一个 EventTouch 对象，我们可以在 ccTouch中使用 API 获取包含在这个对象中的信息，其中包含当前鼠标所在的坐标数据。

我们需要编写一个名为 “initEventListener” 的函数，它可以监听关键的系统事件，并在事件回调中操作游戏内容。在创建函数之后，在 Game.js 的 start 回调中执行一次即可生效。

2.1.1 代码：

```
initEventListener () {
        this.node.on(cc.Node.EventType.MOUSE_MOVE, (event)=>{
            this.onBeCreateHammerEvent(event.getLocation());
        },this);

        this.node.on(cc.Node.EventType.TOUCH_MOVE, (event)=>{
            console.log(event);
            this.onBeCreateHammerEvent(event.getLocation());
        },this);

        this.node.on(cc.Node.EventType.TOUCH_START, (event)=>{
            this.onBeCreateHammerEvent(event.getLocation());
            this.onHammerClicked();
            if (this.gameRuleNode) {
                var gameRuleFadeOut = cc.fadeOut(1);
                this.gameRuleNode.runAction(gameRuleFadeOut);
            }
        },this);

        this.node.on(cc.Node.EventType.TOUCH_END, (event)=>{
            this.onHammerClicked();
        },this);
```

```
}
```

2.2 创造锤子

我们创建一个名为 “onBeCreateHammerEvent” 的函数来生成锤子。让我们首先在场景中创建一个 Sp Hammer 节点，并给它添加 Sprite 组件，以渲染 `hammer.png` 图片。添加完成之后，将其拖到 assets 文件夹下就可以形成 Prefab。当我们让鼠标进入游戏屏幕时，我们将 Sp Hammer 预置体实例化到游戏中，成为我们需要的锤子，它将与我们的鼠标一起移动。

2.2.1 代码：

```
onBeCreateHammerEvent (position) {
     if (!this.hammerNode && !cc.isValid(this.hammerNode)) {
         this.hammerNode = cc.instantiate(this.hammer);
         this.hammerNode.zIndex = cc.macro.MAX_ZINDEX;
         this.hammerNode._isCollider = false;
         this.node.addChild(this.hammerNode);
     }
     this.hammerNode.position = this.node.convertToNodeSpaceAR(position);
},
```

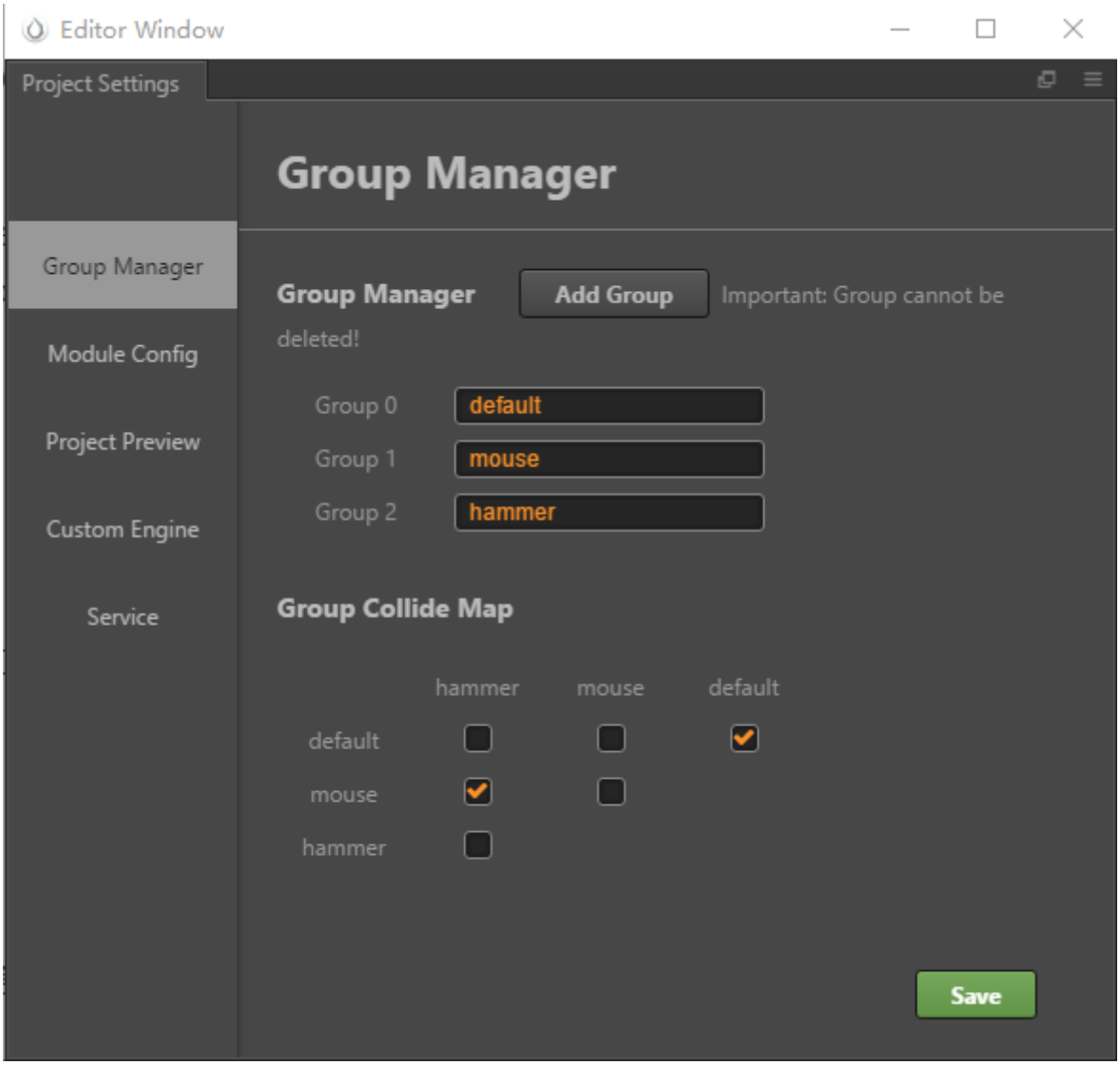
---

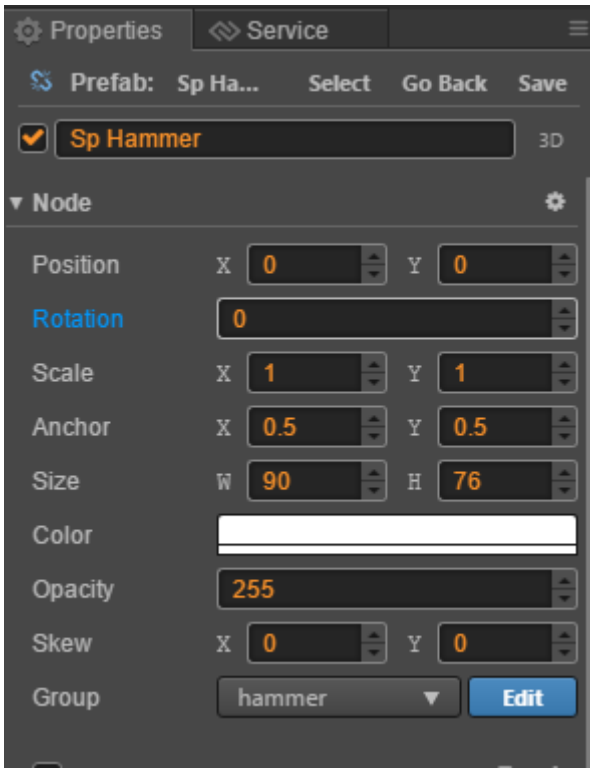
3 使用碰撞系统

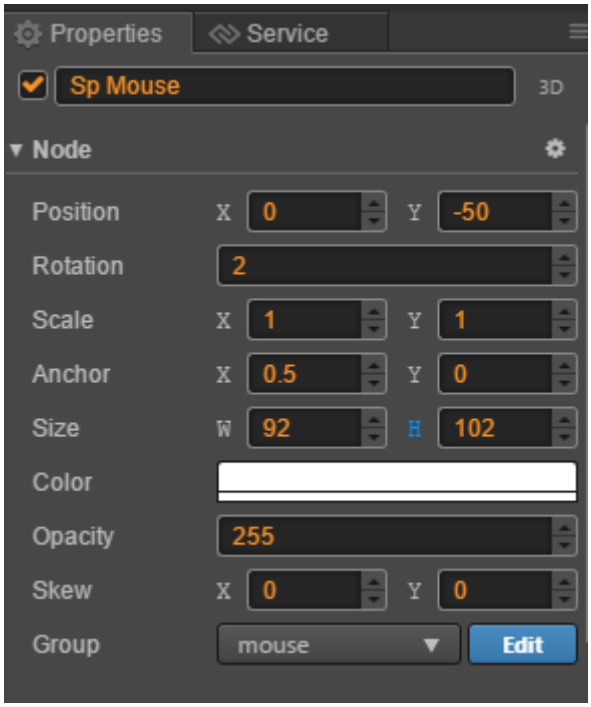
我们希望当鼠标碰到老鼠时，按下鼠标击晕老鼠，而当鼠标没碰到老鼠时则不行。我们可以使用碰撞系统来帮助我们快速实现检测鼠标与老鼠是否相遇。

3.1 更改需要碰撞检测的节点的分组并添加碰撞组件

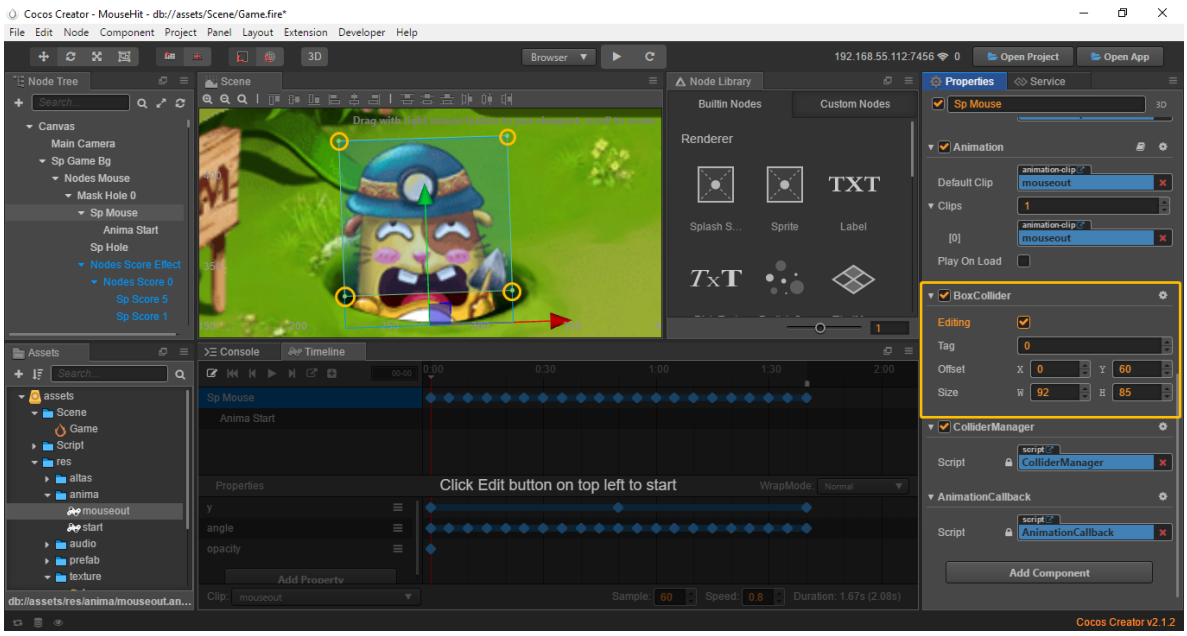
我们需要使用编辑器创建两个新的分组：mouse 和 hammer。菜单路径是 Project -> Project Settings.. -> Group Manager。创建了它之后，我们为需要碰撞检测的节点更改分组。我们将 Sp Mouse 节点的分组更改为mouse，并将 Sp Hammer 的 Prefab 更改为 Hammer。


Editor Window
Project Settings
Group Manager
Group Manager
Module Config
Project Preview
Custom Engine
Service
Group Manager
Add Group
Important: Group cannot be deleted!
Group 0
default
Group 1
mouse
Group 2
hammer
Group Collide Map
hammer
mouse
default
default
mouse
hammer
Save


Properties
Service
Prefab:
Sp Ha...
Select
Go Back
Save
Sp Hammer
3D
Node
Position
X
0
Y
0
Rotation
0
Scale
X
1
Y
1
Anchor
X
0.5
Y
0.5
Size
W
90
H
76
Color
Opacity
255
Skew
X
0
Y
0
Group
hammer
Edit



我们将BoxCollider组件添加到两个节点，单击`'Editting'`，启动碰撞框编辑，我们可以自由拖动四边点来更改碰撞盒的大小。你可以暂时关闭BoxCollider，在它从鼠洞钻出的时候再开启。



3.2 使用脚本控制碰撞系统

默认情况下，CoosCreator 中的碰撞系统是关闭的，因此我们需要手动启动它。我们还需要在碰撞系统中增加产生碰撞和碰撞结束的回调函数。 因此，我们创建了一个名为 “ColliderManager.js” 的 JS 脚本，并将其添加到节点中。通过访问 cc.director 来获取碰撞管理器，并将其`enabled`的属性更改为`true`，以打开碰撞系统。然后在脚本中添加 ‘onCollisionEnter’，‘onCollisionExit’ 函数，这是碰撞生成和碰撞结束的回调，可以控制游戏的内容。这样简单的碰撞系统就创建完成了。

```
cc.Class({
    extends: cc.Component,

    editor: {
        menu: "Custom component / collision management",
    },
```

```
    properties: {
    },

    onLoad () {
        var colliderManager = cc.director.getCollisionManager();
        colliderManager.enabled = true;
        // colliderManager.enabledDebugDraw = true;
    },

    start () {
        this.gameComponent =
cc.director.getScene().getChildByName("Canvas").getComponent("Game");
    },

    onCollisionEnter (other, self) {
        if (this.node.group === cc.game.groupList[1]) {
            this.node._isCollider = true;
            this.gameComponent._mouseNode = this.node;
        }
        else if (this.node.group === cc.game.groupList[2]) {

        }
    },

    onCollisionExit (other, self) {
        if (this.node.group === cc.game.groupList[1]) {
            this.node._isCollider = false;
        }
        else if (this.node.group === cc.game.groupList[2]) {

        }
    },
});
```

4、完善游戏逻辑

我们需要继续补充游戏的逻辑。

4.1 让鼠标随意出现

目前，游戏中有九个鼠洞。我们首先要确定这段时间会有多少老鼠出现，这些老鼠会随机在哪个鼠洞出现，出现的老鼠会随机是哪种类型的，是什么样的老鼠？一旦确定了这个想法，我们就可以设计代码。在创建函数之后，在 Game.js 的 start 回调中执行一次即可生效。

4.1.1 代码：

```
initMouseOutEvent () {
        if (this._mouseIndexArr.length === 0) {
            let mouseAmount = Math.floor((Math.random() *
(this.mouseNodes.childrenCount - 1) + 1));

            if (mouseAmount === 0) {
                mouseAmount = 1;
```

```
            }

            for (let i = 0; i < mouseAmount; i++) {
                let randomNodeIndex = Math.floor(Math.random() *
(this.mouseNodes.childrenCount));
                let randomSpriteFrameIndex = Math.floor(Math.random() *
(this.animalAtlas.getSpriteFrames().length - 1))

                if (this._mouseIndexArr.indexOf(randomNodeIndex) === -1) {
                    var mouseNode =
this.mouseNodes.children[randomNodeIndex].getChildByName("Sp Mouse");
                    this.updateMouseNodeInfo(mouseNode, randomSpriteFrameIndex);
                    mouseNode.getComponent(cc.BoxCollider).enabled = true;
                    this._mouseIndexArr.push(randomNodeIndex);
                    mouseNode.getComponent(cc.Sprite).spriteFrame =
this.animalAtlas.getSpriteFrames()[randomSpriteFrameIndex];
                    mouseNode.getComponent(cc.Animation).play();
                }
            }
        }
    },
```

4.2 我们需要让鼠标重新出现

老鼠出现了一次，然后又进了老鼠洞。我们不得不让老鼠钻进老鼠洞后，从不同的老鼠洞里钻出来。因为我们使用动画组件来让老鼠进出老鼠洞，我们可以给动画组件中添加一个监听，当动画结束时，即当老鼠进入老鼠洞时，重新随机出现。为了让老鼠成群结队的出现，我们使用 _mouseIndexArr 数组记录每个随机出现老鼠的老鼠洞的序号。当鼠标回到洞中，序列才会被删除。当所有老鼠都回到老鼠洞之后，我们再让老鼠们重新出现。

首先，添加一个动画结束时的回调，名为 'onAnimationFinishEvent' ：

4.2.1 代码:

```
onAnimationFinishEvent () {
     this._mouseIndexArr.pop();
     this.initMouseOutEvent();
},
```

接下来，在 "initEventListener" 中添加动画侦听:

```
for (let i = 0; i < this.mouseNodes.childrenCount; i++) {
     this.mouseNodes.children[i].getChildByName("Sp Mouse").
         getComponent(cc.Animation).on(cc.Animation.EventType.FINISHED,
     this.onAnimationFinishEvent, this);
}
```

4.3 锤子下锤的逻辑与表达

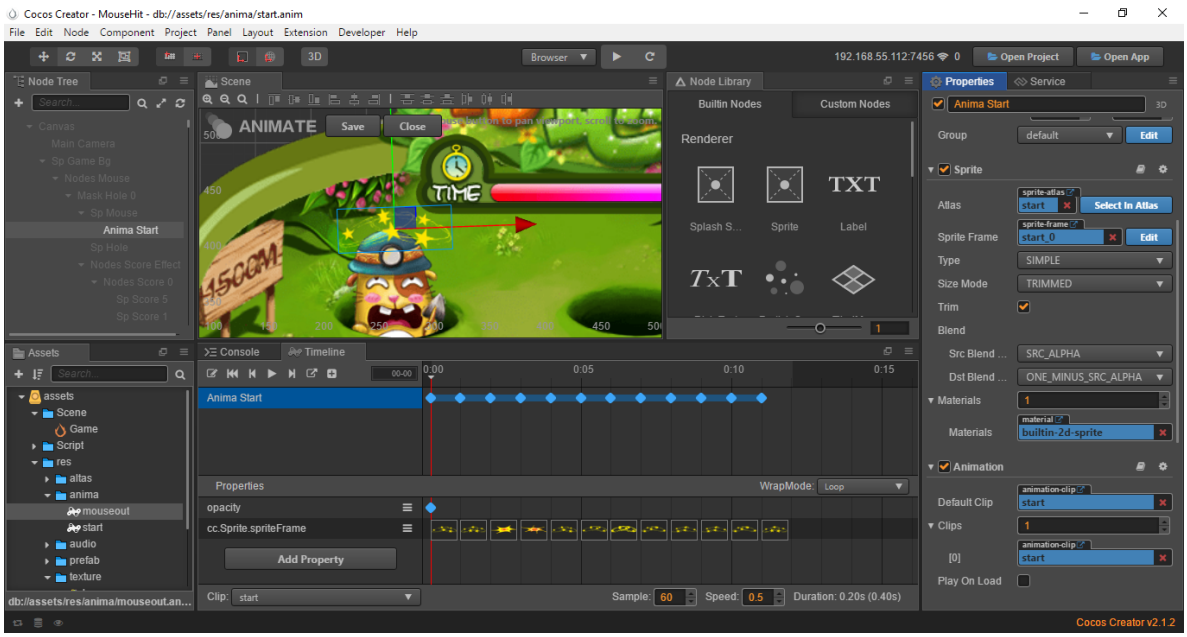
当锤子下降时，锤子会上下翻飞，所以我们需要改变它的 angle 属性来达到这个效果。当锤子下捶时，老鼠就会被击倒。而当老鼠已经被击晕之后，就不能再被重复击晕了。

4.3.1 代码：

```
onHammerClicked () {
        this.hammerNode.angle = this.hammerNode.angle === 0 ? 30 : 0;
        if (this._mouseNode && this._mouseNode._isCollider &&
this._mouseNode._isLive && cc.find("Canvas/Sp Game Bg")) {
            this._mouseNode._scoreUpdateFunc();
            this.showScoreEffectByTag(this._mouseNode,
this._mouseNode.parent.getChildByName("Nodes Score Effect"));
            this.gameScore.string = this._score;
            this._mouseNode._isLive = false;
            let oldSpriteFrameName =
this._mouseNode.getComponent(cc.Sprite).spriteFrame.name;
            let newSpriteFrameName = oldSpriteFrameName + "_death";
            this._mouseNode.getComponent(cc.Sprite).spriteFrame =
this.animalDeathAtlas.getSpriteFrame(newSpriteFrameName);
            this._mouseNode.getChildByName("Anima
Start").getComponent(cc.Animation).play();
        }
}
```

要想在老鼠被击晕时增加动画效果，我们需要使用动画组件。首先，为 Sp Mouse 创建一个新的 Anima Start 节点，给它添加 Animation 组件，并在 assets 文件夹下创建一个新的 AnimationClip，命名为 “start”，它将会重复播放，直到老鼠回到鼠洞中。

'start' 动画需要使用一组图片来显示眩晕效果，我们需要控制节点 Sprite 组件中的 SpriteFrame 属性。所以我们需要添加多个动画帧并对每个动画帧设置不同的 SpriteFrame。



5、写在结尾

本教程主要介绍仓鼠游戏的核心游戏设计，在下一篇文章中将对游戏分数、游戏声音效果、游戏外围交互等进行补充。

谢谢

新项目的**GitHub**链接

完整项目的**GitHub**链接