链滴

python 生成 cocos 序列帧动画

作者：a610569731

原文链接：https://ld246.com/article/1545286769467

来源网站：链滴

散图自动生成序列帧anim动画文件

<!--more-->

配置文件:

config.txt:

```
{
    "path":"/Users/wangankang/Desktop/ani",
    "time":1.2,
    "wrapMode":"1",
    "speed":1,
    "sample": 60
}
```

序列帧散图

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*

import os
import json

class AniTemplate:
    spriteFrame = []
    events = []
    name = ""
    duration = 1
    wrapMode = "1"
    speed = 1
    sample = 60

    def __init__(self, name, time1):
        self.name = name
        self.duration = time1

    def setWrapMode(self, wrapMode):
        self.wrapMode = wrapMode

    def setSpeed(self, speed):
        self.speed = speed

    def setSample(self, sample):
        self.sample = sample

    def obj_2_json(self):
        return {
            "__type__": "cc.AnimationClip",
            "_name": self.name,
            "_objFlags": 0,
            "_duration": self.duration,
```

```
            "sample": self.sample,
            "speed": self.speed,
            "wrapMode": self.wrapMode,
            "curveData": {
                "comps": {
                    "cc.Sprite": {
                        "spriteFrame": self.spriteFrame
                    }
                }
            },
            "events": self.events
        }



# 读取一个json文件,并且转为json对象
def fileToJson(fileName):
    file = open(fileName)
    str = file.read()
    file.close()
    return json.loads(str)

# 读取一个meta文件中的uuid
def getMetaUuid(path):
    basename = os.path.basename(path)
    meta = fileToJson(path)
    basename = basename[0:basename.rfind(".")]
    basename = basename[0:basename.rfind(".")]
    return meta['subMetas'][basename]['uuid']

def saveStrToFile(file_name, contents):
    fh = open(file_name, 'w')
    fh.write(contents)
    fh.close()


def genAnim(path, name, time,wrapMode,speed, sample, has_even):
    print("==================================")

    fileList = os.listdir(path)
    metaList = []
    for i in range(0, len(fileList)):
        if fileList[i].endswith(".meta"):
            metaList.append(fileList[i])

    metaList.sort()

    print("读取meta序列帧文件完成,共{" + str(len(metaList)) + "}个")

    if 0 == len(metaList):
        print(name + ">没有找到meta文件,无法生成序列帧,自动跳过")
        return

    # step = time / (len(metaList) - 1)
```

```
    step = 0.125
    time = 0.125 * len(metaList)

    print("序列帧动画间隔为:" + str(step) + "s")

    print("准备生成动画文件....")
    template = AniTemplate(name, time);

    template.setSample(sample)
    template.setSpeed(speed)
    template.setWrapMode(wrapMode)

    lastFrameTime = 0

    spriteFrame = []
    for i in range(0, len(metaList)):
        lastFrameTime = i*step
        spriteFrame.append({
            "frame": i * step,
            "value": {
                "__uuid__": getMetaUuid(path +"/" + metaList[i])
            }
        })

    template.spriteFrame = spriteFrame;

    if has_even:
        template.events = [{
            "frame": lastFrameTime,
            "func": "actionOver",
            "params": []
        }]
    else:
        template.events = []

    saveStrToFile(name + '.anim', json.dumps(template, default=AniTemplate.obj_2_json, sort_
eys=False, indent=4))

    print("文件生成完成:\n" + os.path.realpath(name + '.anim'))

#
# ========================================
#            main
# ========================================
#

print("开始读取config配置数据...")
config = fileToJson("./config.txt")
print("读取config配置成功...")

path = config['path']
time = config['time']
# wrapMode = config['wrapMode']
speed = config['speed']
```

```
sample = config['sample']

print("配置数据解析完成")
print("\tpath:" + path)
print("\ttime:" + str(time))

root = os.listdir(path)
for i in range(0, len(root)):
    if os.path.isdir(path + "/" + root[i]):
        aniPath = path + "/" + root[i]
        aniName = root[i]
        aniName = str(aniName)
        hasAtk = aniName.find("Attack") != -1
        hasStand = aniName.find("Stand") != -1

        hasEvent = hasAtk

        wrapMode = 1
        if hasStand:
            wrapMode = 2

        genAnim(aniPath, aniName, time,wrapMode,speed, sample, hasEvent)
```

序列帧合集:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*

import os
import json

class AniTemplate:
    spriteFrame = []
    events = []
    name = ""
    duration = 1
    wrapMode = "1"
    speed = 1
    sample = 60

    def __init__(self, name, time1):
        self.name = name
        self.duration = time1

    def setWrapMode(self, wrapMode):
        self.wrapMode = wrapMode

    def setSpeed(self, speed):
        self.speed = speed

    def setSample(self, sample):
```

```
        self.sample = sample

    def obj_2_json(self):
        return {
            "__type__": "cc.AnimationClip",
            "_name": self.name,
            "_objFlags": 0,
            "_duration": self.duration,
            "sample": self.sample,
            "speed": self.speed,
            "wrapMode": self.wrapMode,
            "curveData": {
                "comps": {
                    "cc.Sprite": {
                        "spriteFrame": self.spriteFrame
                    }
                }
            },
            "events": self.events
        }



# 读取一个json文件,并且转为json对象
def fileToJson(fileName):
    file = open(fileName)
    str = file.read()
    file.close()
    return json.loads(str)

# 读取一个meta文件中的uuid
def getMetaUuid(path):
    basename = os.path.basename(path)
    meta = fileToJson(path)
    basename = basename[0:basename.rfind(".")]
    basename = basename[0:basename.rfind(".")]
    return meta['subMetas'][basename]['uuid']


def get_plist_uuid(path):
    basename = os.path.basename(path);
    json = fileToJson(path)
    sub_metas = json['subMetas']

    key_list = [];

    for key in sub_metas:
        key_list.append(key)

    # 自定义排序 0.png => int("0.png".sub(0,xxx.indexOf("."))) => int("0)
    key_list.sort(key=lambda i: int(i[0:i.rfind(".")]), reverse=False)


    uuid_list = []
```

```
    for i in range(0, len(key_list)):
        uuid = json['subMetas'][key_list[i]]['uuid']
        uuid_list.append(uuid)
    return uuid_list;

def saveStrToFile(file_name, contents):
    fh = open(file_name, 'w')
    fh.write(contents)
    fh.close()

def genAnim(path, name, time,wrapMode,speed, sample, has_even):
    print("================================")

    fileList = os.listdir(path)
    metaList = []
    for i in range(0, len(fileList)):
        if fileList[i].endswith(".plist.meta"):
            metaList.append(fileList[i])

    metaList.sort()

    print("读取当前文件夹meta plist文件完成,共{" + str(len(metaList)) + "}个plist")

    # 获取plist 里面的所有uuid
    uuid_list = get_plist_uuid(path +"/" + metaList[0])

    if 0 == len(metaList):
        print(name + ">没有找到meta文件,无法生成序列帧,自动跳过")
        return

    if 0 == len(uuid_list):
        print(name + ">没有找到plist下的uuid, 无法生成序列帧动画,自动跳过")
        return

    # step = time / (len(metaList) - 1)
    step = 0.125
    # time = 0.125 * len(metaList)
    time = 0.125 * len(uuid_list)

    print("序列帧动画间隔为:" + str(step) + "s")

    print("准备生成动画文件....")
    template = AniTemplate(name, time);

    template.setSample(sample)
    template.setSpeed(speed)
    template.setWrapMode(wrapMode)

    lastFrameTime = 0

    spriteFrame = []
    for i in range(0, len(uuid_list)):
        lastFrameTime = i*step
        spriteFrame.append({
```

```
            "frame": i * step,
            "value": {
                # "__uuid__": getMetaUuid(path +"/" + metaList[i])
                "__uuid__": uuid_list[i]
            }
        })

    template.spriteFrame = spriteFrame;

    if has_even:
        template.events = [{
            "frame": lastFrameTime,
            "func": "actionOver",
            "params": []
        }]
    else:
        template.events = []

    saveStrToFile(name + '.anim', json.dumps(template, default=AniTemplate.obj_2_json, sort_
eys=False, indent=4))

    print("文件生成完成:\n" + os.path.realpath(name + '.anim'))

#
# ========================================
#             main
# ========================================
#

print("开始读取config配置数据...")
config = fileToJson("./config.txt")
print("读取config配置成功...")

path = config['path']
time = config['time']
# wrapMode = config['wrapMode']
speed = config['speed']
sample = config['sample']

print("配置数据解析完成")
print("\tpath:" + path)
print("\ttime:" + str(time))

root = os.listdir(path)
for i in range(0, len(root)):
    if os.path.isdir(path + "/" + root[i]):
        aniPath = path + "/" + root[i]
        aniName = root[i]
        aniName = str(aniName)
        hasAtk = aniName.find("Attack") != -1
        hasStand = aniName.find("Stand") != -1

        hasEvent = hasAtk
```

```
wrapMode = 1
if hasStand:
    wrapMode = 2

genAnim(aniPath, aniName, time,wrapMode,speed, sample, hasEvent)
```