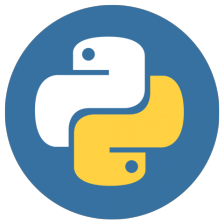
**如何用OpenGL的点精灵（point sprite）绘制雪花？**

[[](https://blog.csdn.net/xufive/category_6727443.html)](https://blog.csdn.net/xufive/category_6727443.html)

看冬奥才知道，阿勒泰不但是中国的“雪都”，还是“人类[滑雪](https://so.csdn.net/so/search?q=%E6%BB%91%E9%9B%AA&spm=1001.2101.3001.7020)起源地”。这个说法是否成立，姑且不论，阿勒泰的雪的确很漂亮。冬奥会有一个宣传片，就是借用一朵阿勒泰雪花的视角来讲述冬奥会的故事，既有历史的厚重，又有艺术的浪漫，极具视觉冲击感。

  
那么问题来了：如何用[OpenGL](https://so.csdn.net/so/search?q=OpenGL&spm=1001.2101.3001.7020)绘制雪花呢？通常，点精灵（point sprite）技术被用于描述大量粒子在屏幕上的运动，自然也可以用于绘制雪花。点精灵可以理解为贴了纹理图片的点——仅用一个vertex就可以把一个2D纹理图片绘制到屏幕的任何位置。

在OpenGL中开启和使用点精灵有一点点复杂，好在WxGL对此做了封装，用起来非常简单。在给出演示代码前，先贴两张雪花的[纹理](https://so.csdn.net/so/search?q=%E7%BA%B9%E7%90%86&spm=1001.2101.3001.7020)图片。

* snow\_1.png  
  
* snow\_2.png  
  

熟悉GLSL语言的同学，很容易读懂着色器源码。将着色器源码、纹理图片装进模型之后，只需要show一下，雪花就显示出来了。如果想实现雪花飘飘的效果，请参考我的另一篇博文[《用OpenGL导演一场烟花盛会，迎接即将到来的新年》](https://xufive.blog.csdn.net/article/details/122743824)。

import numpy as np

import wxgl

from wxgl import wxplot as plt

vshader\_src = """

#version 330 core

in vec4 a\_Position;

uniform mat4 u\_MVPMatrix;

void main() {

gl\_Position = u\_MVPMatrix \* a\_Position;

gl\_PointSize = (a\_Position.z + 1) \* 30;

}

"""

fshader\_src = """

#version 330 core

uniform sampler2D u\_Snow\_1;

in float idx;

void main() {

gl\_FragColor = texture2D(u\_Snow\_1, gl\_PointCoord);

}

"""

m = wxgl.Model(wxgl.POINTS, vshader\_src, fshader\_src, sprite=True) # 通过sprite=Treue开启点精灵

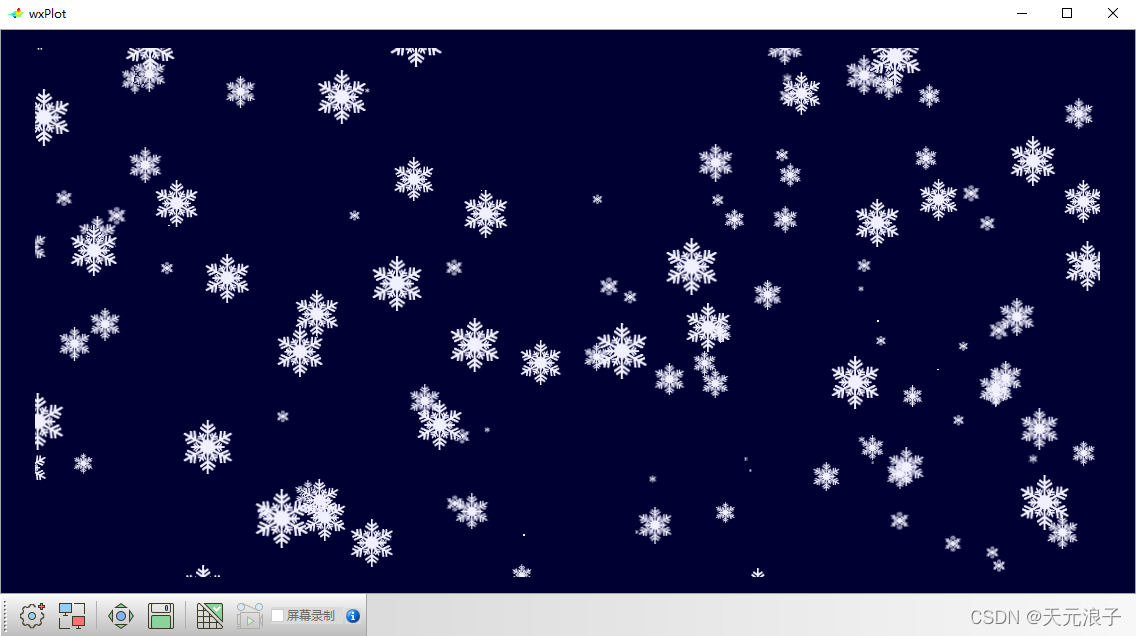
m.set\_vertex('a\_Position', np.random.random((300, 3))\*2-1) # 随机生成300个点

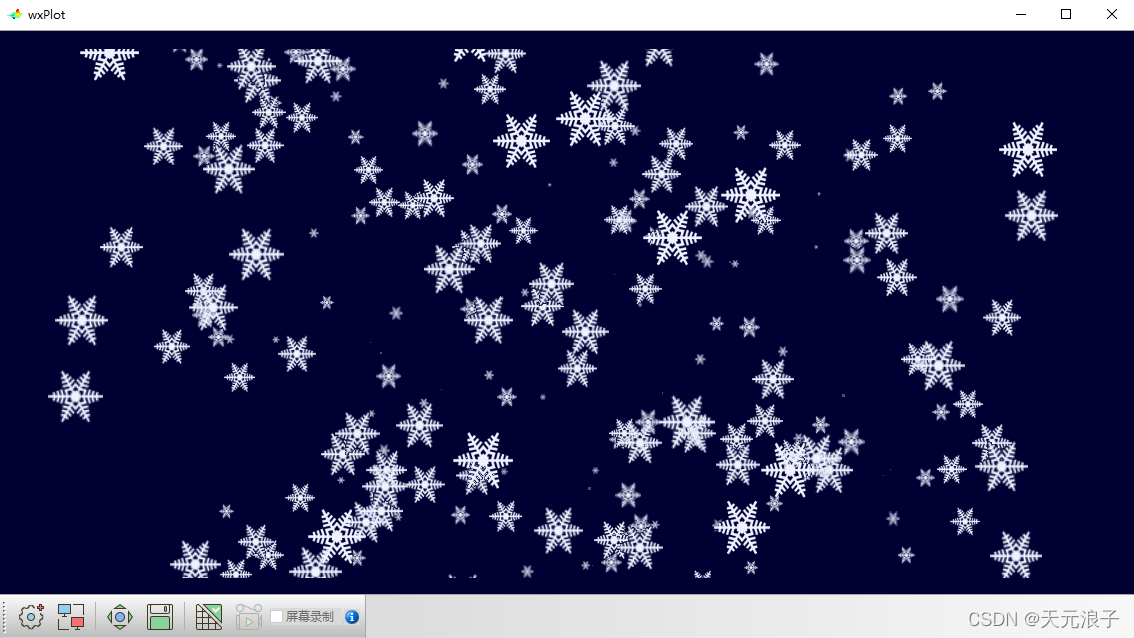
m.add\_texture('u\_Snow\_1', 'res/image/snow\_1.png', wxgl.TEXTURE\_2D) # 添加雪花纹理

m.set\_mvp\_matrix('u\_MVPMatrix') # 设置模型矩阵、视点矩阵和投影矩阵

plt.model(m)

plt.show()

下面是使用snow\_1.png做纹理的效果。  


下面是使用snow\_2.png做纹理的效果。  
  
不过，这样的雪花略显单调，毕竟，世界上没有两片完全相同的雪花。怎样让雪花看起来更逼真一点呢？下面的代码尝试在片元着色器中混用两种纹理。

import numpy as np

import wxgl

from wxgl import wxplot as plt

vshader\_src = """

#version 330 core

in vec4 a\_Position;

uniform mat4 u\_MVPMatrix;

void main() {

gl\_Position = u\_MVPMatrix \* a\_Position;

gl\_PointSize = (a\_Position.z + 1) \* 30;

}

"""

fshader\_src = """

#version 330 core

uniform sampler2D u\_Snow\_1;

uniform sampler2D u\_Snow\_2;

in float idx;

void main() {

if (fract(sin(dot(gl\_PointCoord ,vec2(12.9898,78.233))) \* 43758.5453) < 0.5) {

gl\_FragColor = texture2D(u\_Snow\_1, gl\_PointCoord);

} else {

gl\_FragColor = texture2D(u\_Snow\_2, gl\_PointCoord);

}

}

"""

m = wxgl.Model(wxgl.POINTS, vshader\_src, fshader\_src, sprite=True) # 通过sprite=Treue开启点精灵

m.set\_vertex('a\_Position', np.random.random((300, 3))\*2-1) # 随机生成300个点

m.add\_texture('u\_Snow\_1', 'res/image/snow\_1.png', wxgl.TEXTURE\_2D) # 添加雪花纹理1

m.add\_texture('u\_Snow\_2', 'res/image/snow\_2.png', wxgl.TEXTURE\_2D) # 添加雪花纹理2

m.set\_mvp\_matrix('u\_MVPMatrix') # 设置模型矩阵、视点矩阵和投影矩阵

plt.model(m)

plt.show()

这个雪花有点独特吧？