

文章编号: 2096-1618(2025)01-0014-07

一种基于柏林噪声和分形布朗运动的过程纹理合成方法

李茄濡, 何晓曦, 刘应浒, 孟繁林, 朱群

(成都信息工程大学软件工程学院, 四川 成都 610225)

摘要:针对现有的人工智能生成纹理和基于样图的纹理合成中存在的不可控、缺乏灵活性、低实时性等问题,运用过程纹理生成技术,提出一种基于柏林噪声和分形布朗运动的纹理生成算法,用于合成真实木制纹理特征。该算法首先将待生成的纹理空间划分为规则且均匀的网格点,同时在每个网格顶点处随机生成一个梯度向量,然后使用三线性插值方法对网格点上的梯度向量进行插值运算,同时将多频率、多振幅的噪声纹通过分形算法加权叠加处理,生成一个平滑的纹理图案,在此基础上对纹理进行缩放、扭曲、添加木眼来模拟真实的纹理结构。实验证明该算法能真实地模拟木制纹理的特征,且具有实时、高效、可定制性强等优点,在游戏、建模、虚拟现实等领域具有广泛的应用价值。

关键词:过程纹理;柏林噪声;分形;插值;实时

中图分类号:TP391

文献标志码:A

doi:10.16836/j.cnki.jcuit.2025.01.003

0 引言

现有的纹理合成技术主要有三大类:基于人工智能的纹理合成、基于样图的纹理合成和过程化纹理生成方法。

随着深度学习的快速发展,许多基于人工智能的纹理生成方法取得了重要进展。Gatys等^[1]提出一种基于卷积神经网络特征空间的自然纹理生成模型,能产生与非参数纹理合成方法质量相当的自然纹理,但合成过程需要耗费大量时间。为解决Gatys计算方法耗时的问题,Ulyanov等^[2]提出一种代替方法,将生成计算过程中的负担转移到学习阶段,通过训练前馈卷积网络来生成任意大小的纹理。Zhou等^[3]通过单幅纹理图像上过拟合生成对抗网络,实现非均匀纹理的扩展合成。虽然合成结果质量很高,但该方法一方面受网络结构限制,只能合成输入纹理2倍大小的结果,不能合成任意高宽比或任意大小的非均匀纹理,无法适应多变的纹理合成需求;另一方面,该方法也没有提供方便直观的引导图,让用户来控制纹理的合成结果。陈凯健等^[4]针对此问题提出一种基于相对坐标控制的非均匀纹理合成方法,引入相对坐标信息作为条件输入引导控制合成,同时通过下采样和角度旋转,提升纹理质量。但基于人工智能的纹理合成方法都需要准备大量数据集进行长时间训练,且模型生成纹理质量

和数据集的质量相关。

基于样图的纹理生成方法一般需要用户提供一张输入图片或几何图形,然后根据不同的需求选择适用的生成方法。Efors等^[5]提出基于块的纹理合成算法,保证合成过程中块边缘的纹理信息合理性,但合成过程中会指定一个领域大小,可能会导致不均匀分布的模式分布,同时在生成大尺寸纹理时需要更多的计算资源。徐晓刚等^[6]基于相关性原理利用双向扫描算法提高了纹理合成的速度同时可以根据需求生成新的纹理,但是对结构性纹理进行处理时还存在效果较差的问题。而靳利贞等^[7]提出基于双接缝线一致性准则的子块搜索策略和非重叠拼接算法,虽然提高了基于块的纹理合成算法的质量和速度,但是并不能做到实时的纹理生成。以上的纹理合成方法都需要提供一张样图,并不能满足在没有样图的情况下合成纹理,同时用户对纹理的可操作空间狭窄。

过程纹理则通过使用算法、数学模型或物理模拟等手段,实现从无到有生成纹理的细节和特征,以达到更高的灵活性和自由度。为处理物体表面的纹理细节,Peachey^[8]提出一个概念即立体纹理函数,使用在三维空间区域中定义的纹理函数,许多非均匀的材料(包括木材和石头),在三维立体表面更真实地呈现,同时允许用户通过定义纹理函数和参数来控制纹理的外观和特性;Perlin^[9]在此基础上加入噪声函数,能很好地模仿大理石、湍流等纹理。对于充满艺术感、拥有重复、递归的纹理图案,Santoni等^[10]提出将文法作用于纹理生成,给用户提供了3种不同的操作符用于处理纹理,可以对目标区域做分割,填充不同的图形及对目

收稿日期:2023-09-07

基金项目:四川省科技厅重大专项资助项目(2022ZDZX0001);四川省科技厅重点研发资助项目(2022YFG0033、2022YFG0037);四川省信息化应用支撑软件工程技术研究中心开放课题项目(2021RJGC-Y01)

通信作者:何晓曦, Email:microwest@cuit.edu.cn

标区域进变形,使纹理内容更加丰富,拥有一定的可定制性。Liu 等^[11]通过纹理的感知属性,设计一种基于实例生成新过程纹理的框架,通过外界输入一定的感知属性参数,在纹理空间中进行感知相似度度量来确定过程纹理生成模型的参数。但这种方法需要预先准备对各种纹理的感知数据,且会造成感知空间样本稀疏的问题。Larsson 等^[12]提出一种用于合成实木年轮的程序框架,虽然合成的年轮纹理符合自然生长状况,但是此方法对年轮的大小、圈数、角度任意改变,此外对于实木的表皮结构并不能进行模拟。

在柏林噪声方面,王志强等^[13]通过二维柏林噪声控制花瓣和叶片的卷曲程度,实现花卉器官的三维重建。陈国栋等^[14]以柏林噪声为基础,提出可用于细化心脏纹理图的新方法,可对心脏客观结构进行较好的模拟。霍星等^[15]使用正弦函数对加权的一维 Perlin 和二维 Perlin 函数进行扰动,成功进行了木纹的仿真效果。但是,此方法对纹理的扭曲程度是由当前点的正弦值决定的,不能由外界控制,同时在木制表面的纹理合成还存在空缺。

基于此,本文提出一种高效、简单的纹理扭曲方法和一种绘制纹理表面木眼的方法,并通过柏林噪声函数和 FBM 算法进行木制纹理生成,这种生成方法可通过 GPU 实时生成纹理,实现合成高效、纹理真实、使用方便、参数可控的纹理合成技术。

1 木制纹理生成算法

为合成扭曲的木制纹理,通过生成分形柏林噪声,并对噪声纹理进行缩放、扭曲以获得木制纹理的表面纹理,然后对纹理进行添加木眼结构,使其更加接近真实的木纹,然后对纹理使用配色表映射进行纹理着色,最后输出纹理贴图。

1.1 分形噪声

柏林噪声函数的生成主要依靠噪声函数和插值函数。设 p 为输入点的坐标, m, n, k, q 为互不相等且不等于 0 的任意浮点数, $seed$ 为随机种子,噪声函数可以定义为

$$\begin{cases} \text{randomVec} = \text{float2}(\text{dot}(p, (m, n)), \text{dot}(p, (k, q))) \\ \text{randomValue} = -1 + 2 \cdot \text{frac}(\sin(\text{randomVec}) \cdot \text{seed}) \end{cases} \quad (1)$$

为保证产生的随机向量具有一定的随机性,通过不相等的浮点数和坐标进行点乘操作可以较大地获得随机值, randomValue 则是最终被映射到 $[-1, 1]$ 之后的值。根据上述函数,输入坐标然后返回一个随

机数,并且可以通过调节不同的随机种子生成不同的噪声值。而插值函数则是通过对随机产生的点进行光滑处理,使点与点之间的过度变得平滑,从而将噪声纹理整体变得更加合理。

$$\text{lerp}(x, y, \text{weight}) = x + (y - x) \cdot \text{weight} \quad (2)$$

式中, x, y 为需要进行插值的数值, weight 则是控制点,用于控制 x, y 所占的比重。本文柏林噪声的生成过程包括网格点和随机向量的生成、计算相对位置权重、插值叠加。对于一张待生成的纹理,先将其划分为均匀且规则的正方形网格点,然后在每个正方形网格点的顶点位置根据随机噪声函数生成一个由随机值组成的向量。通过每个像素点坐标减去当前像素所在方格顶点的向量,得到顶点到中心点的梯度向量,通过点乘计算每个梯度向量在当前位置坐标上的投影作为各个顶点的噪声值,然后使用权重公式计算各个噪声值在 u, v 方向上的权重,最后根据使用三线性插值函数在顶点噪声之间进行插值操作。

噪声生成过程定义为

$$\text{value} = \text{Noise}(x, y) \quad (3)$$

式中 x, y 代表当前像素点的 uv 坐标, value 代表此点生成的噪声值。常用加权方式有两种^[16],其中 p 代表 uv 坐标位置, $f(p)$ 是经过加权公式计算后的线性插值参数。

$$f(p) = 3p^2 - 2p^3 \quad (4)$$

$$f(p) = 6p^5 - 15p^4 + 10p^3 \quad (5)$$

由于式(4)的二阶导数不为 0,这可能导致噪声的不连续性,而式(5)刚好弥补这一缺点,因此本文采用式(5)。

由于上述直接生成的噪声纹理细节不自然,并不能直接用于木制纹理的合成,因此需要在此基础上使用 FBM 算法添加更多的纹理细节,使其看起来更加真实和具有层次感。

FBM 算法的主要思想是通过叠加多个不同尺度和振幅的噪声函数或者纹理来生成更复杂的噪声函数或纹理。多个不同频率和振幅的一维柏林噪声图像见图 1。

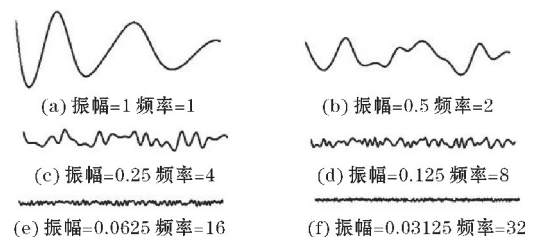


图1 一维噪声波形图

基于 FBM 算法,通过对上述不同振幅和频率的噪声函数进行叠加处理得到叠加后的一维噪声函数,可

以将叠加后的噪声函数想象成山峰的高低起伏、木制纹理的扭曲、石头的凸起凹陷等。其叠加结果见图2。

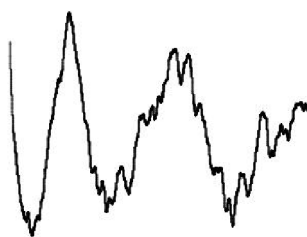


图2 一维噪声叠加结果

在二维纹理图像中,同样可以通过叠加多个不同频率和不同振幅的噪声图像来获取更具细节的纹理图像。纹理的振幅可以直接通过对噪声值进行改变,频率则可以通过对噪声纹理的采样坐标缩放进行改变。

频率和振幅的叠加规律定义为

$$\text{frequency}_{i+1} = 2^i \cdot \text{frequency}_i \quad (6)$$

$$\text{amplitude}_{i+1} = 0.5^i \cdot \text{amplitude}_i \quad (7)$$

式中, i 代表当前叠加次数, frequency_i 代表第 i 次的频率, amplitude_i 代表第 i 次的振幅; frequency 、 amplitude 初始化时都为1。

根据式(6)、(7),二维柏林噪声使用FBM算法的公式如下,其中 n 代表迭代的次数。

$$\text{FBMNoise} = \sum_{i=1}^n \text{Amplitude} \cdot \text{Noise}(\text{frequency} \cdot (x, y)) \quad (8)$$

根据上述公式,在二维纹理中,多频率多振幅的纹理叠加过程见图3。

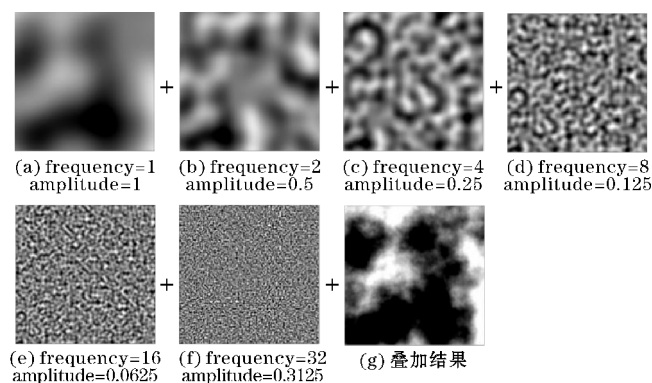


图3 多频率多振幅的二维噪声纹理叠加

可以看出,经过叠加后的噪声细节更多,并且纹理的过度变得平缓,没有那么尖锐,更加自然且具有更多细节。

1.2 噪声缩放

为解决木制纹理在 u 方向上的纹理分布数量,需要在 u 方向上对噪声按照一定的比例进行缩放,进而

改变噪声函数的频率,而在 v 方向上不变。

当在 u 方向的缩放比例等于15时,会大致形成木纹的条状结构,并且伴随着不规则性,见图4。

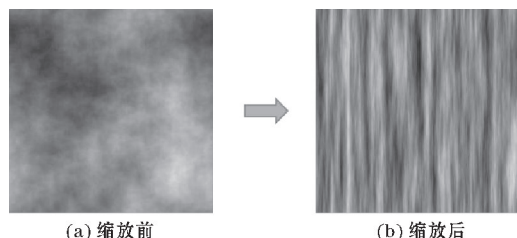


图4 纹理缩放结果

1.3 噪声纹理扭曲算法

通过上述纹理缩放得到的纹理在 v 方向的规律性较强,但自然界中的木制纹理在 v 方向上会呈现一定的随机扭曲。为此,本文提出一种基于灰度图的扭曲算法。

令 θ 代表扭曲的角度大小, $\vec{\text{dir}}$ 代表最终的扭曲方向,则采样偏移公式:

$$\vec{\text{dir}} = \text{float2}(\cos(\theta), \sin(\theta)) \quad (9)$$

式(9)能根据输入的扭曲角度对每个采样点的采样位置进行偏移,具体示例见图5。

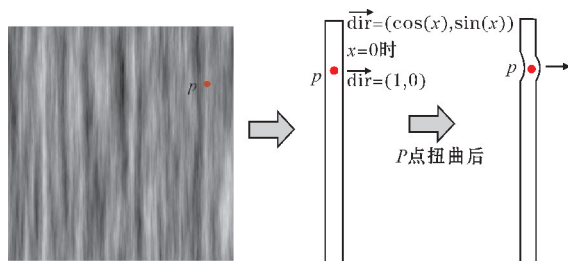


图5 纹理扭曲原理

图5中, p 点是需要扭曲的一个点,当经过扭曲函数计算后,得到该 p 点的最终扭曲方向是 $(1,0)$,即当前 p 点的采样位置更新为 $uv' = uv + (1,0)$ 。

经过上述的偏移,纹理中的所有采样点均会进行一定角度的偏移。但是,由于现实世界中木制纹理的每个点的扭曲程度不一定是相同的,扭曲方向也不一定是一样的,因此本文引入一张同像素大小的灰度图,并且以灰度图的 r 通道值以随机决定每个点的扭曲程度和扭曲方向。灰度图的 r 通道的采样结果范围是 $[0,1]$,为使扭曲程度不要过于夸张且扭曲方向需要有区分,需要使用以下公式将采样结果映射到 $[-0.5,0.5]$,以此作为纹理的扭曲距离。

$$\text{dist} = \text{heightMap}.r - 0.5 \quad (10)$$

式中 dist 的取值范围为 $[-0.5,0.5]$,即灰度图的像素值越接近于0或者1的地方代表扭曲距离越远,采样

时偏移的距离越远,像素值越接近0.5的地方代表扭曲距离越近,采样时偏移的距离越近。

本文的扭曲函数是由外界输入的灰度图、扭曲角度以及扭曲强度决定。灰度图用于控制每个噪声点的基础扭曲程度,扭曲角度用于控制每个点的采样偏移方向,扭曲强度用于加强纹理的扭曲度。

通过以上方法可以得到整个噪声纹理 uv 采样扭曲公式:

$$uv' = uv + \vec{\text{dir}} \cdot \text{dist} \cdot \text{intensity} \quad (11)$$

式中, $\vec{\text{dir}}$ 为输入偏移角度 θ 后的扭曲方向, dist 是根据灰度图计算得到的扭曲距离, intensity 则是输入的扭曲强度。

在扭曲角度为 200, 扭曲强度为 0.09 时最后的噪声纹理扭曲结果见图 6。

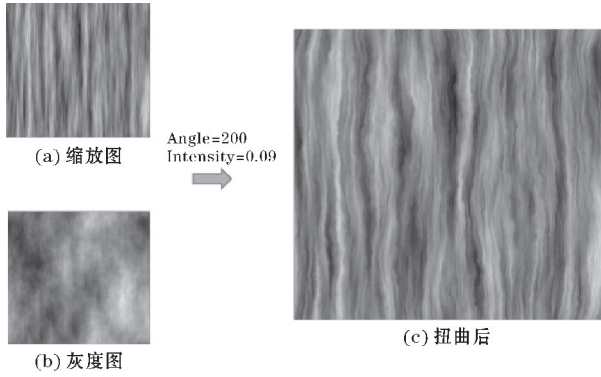


图 6 灰度图对纹理进行扭曲

1.4 绘制木眼

木制纹理的表面可能还存在木眼,类似于椭圆螺旋状,因此可以通过此形状进行木眼的模拟。

上述的纹理扭曲,一个点能够定向在某个角度进行扭曲,而木眼结构需要从一个点同时向四周进行扩散,且在不同的半径内的扭曲程度又有差别,因此在扭曲方向上需要有梯度变化,见图 7。

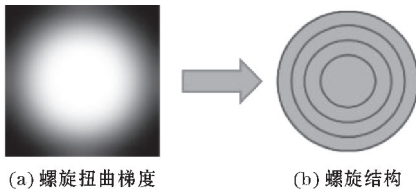


图 7 木眼结构扭曲梯度

同时,为保证木眼边缘的不规则性,不能直接对木纹和螺旋结构进行混合,需要使用梯度螺旋结构纹理图对木纹纹理对应的像素区域进行变形处理,进而达到模拟效果。为此,本文提出一种基于斜率的扭曲算法。

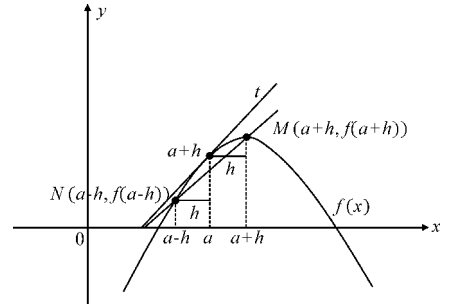


图 8 中心差分法

图 8 是通过中心差分法计算函数 $f(x)$ 计算点 P 的斜率示意图,通过一个小的增量 h 作为步长,向 x 正轴得到点 $M(a+h, f(a+h))$,向 x 负轴得到点 $N(a-h, f(a-h))$,则直线 MN 的斜率为

$$k = (f(a+h) - f(a-h)) / 2h \quad (12)$$

当 h 足够小时, M 、 N 点无限逼近于 P 点,此时 MN 的斜率 k 无限趋近于 P 点的斜率 t ,即可近似求得 P 点的斜率。

当计算二维灰度图像中某个像素点的在 uv 方向上的梯度变化时,以该像素点的位置为中心建立坐标系,分别考虑在 u 、 v 方向上的斜率分布。

具体算法思想是:对当前像素点 (x_i, x_j) 在某个差分步长 step 下,计算在 u 、 v 方向上的偏移值 delta ,然后分别取 $(x_i, x_j + \text{step})$ 、 $(x_i, x_j - \text{step})$ 、 $(x_i + \text{step}, x_j)$ 、 $(x_i - \text{step}, x_j)$ 4 个点,首先计算在 u 方向上的梯度变化,然后计算在 v 方向上的梯度变化,如式 (14),最后根据偏移值和梯度变化进行采样。

因为 u 、 v 的值是在 $0 \sim 1$, step 大小是任意的,而偏移值需要很小且和纹理的大小有关,所以需要在差分步长和偏移值之间做出转换,转换公式如下:

$$\text{delta} = \text{step} / \text{textureSize} \quad (13)$$

delta 代表由输入的差分步计算得到的偏移值, C 代表在经过偏移后的采样点的采样结果, Gradient 代表梯度变化,则得到在 u 、 v 方向上梯度变化:

$$\begin{cases} \text{Gradient}_{u_i} = (C_{(u_i + \text{delta}, v_j)} - C_{(u_i - \text{delta}, v_j)}) / 2 \cdot \text{delta} \\ \text{Gradient}_{v_j} = (C_{(u_i, v_j + \text{delta})} - C_{(u_i, v_j - \text{delta})}) / 2 \cdot \text{delta} \end{cases} \quad (14)$$

当前点总的梯度变化为

$$\vec{\text{Gradient}} = \text{float2}(\text{Gradient}_{u_i}, \text{Gradient}_{v_j}) \quad (15)$$

此外,为使纹理变化强度得到提升,使用一个参数 Intensity 进行控制。算法的具体步骤如下:

基于斜率的扭曲算法 (SDA)

输入: 梯度纹理 GradientTexture , 差分步长 step , 噪声纹理 Noise Texture , 扭曲强度 Intensity 。

输出: 扭曲后的采样结果。

(1) 根据差分步长 step 和纹理大小计算纹理采样偏移值 delta ;

(2)根据偏移值计算偏移过后的4个点: $(x_i, x_j + \text{step})$ 、 $(x_i, x_j - \text{step})$ 、 $(x_i + \text{step}, x_j)$ 、 $(x_i - \text{step}, x_j)$ 并且取出每个点的灰度值;

(3)通过式(14)计算在 u 方向和 v 方向上的灰度值梯度变化率;

(4)计算最终的梯度变化率 $\overrightarrow{\text{Gradient}}$;

(5)根据梯度变化率和偏移强度,完成对主纹理的偏移采样, $\text{color} = \text{tex2D}(\text{GradientTexture}, i.uv + \text{Gradient} \cdot \text{Intensity})$;

(6)返回采样结果。

基于上述算法,纹理经过扭曲后模拟添加木眼结果见图9。

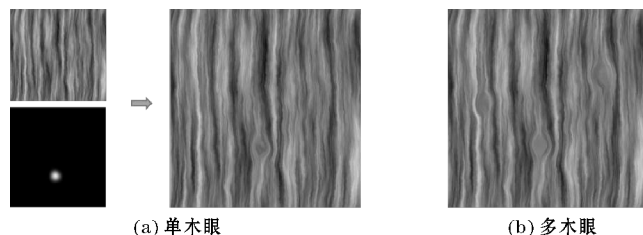


图9 添加木眼结果

1.5 纹理着色

纹理着色时,需要一张配色表,宽度为256个像素,高度任意大小,每个像素值代表最终纹理需要呈现的颜色。由配色表组成的像素矩阵中,列下标相同的元素像素值相同。

设映射表为 T ,其中的元素个数为256个像素值,元素下标0~255;设 $\text{Color}(i, j)$ 为灰度图坐标 (i, j) 的灰度值,则映射过程如下:

$$\text{Color}'(i, j) = T[\text{Color}(i, j)] \quad (16)$$

具体映射过程是对于一张灰度纹理图,读取灰度纹理的灰度值,接着通过配色表映射出当前灰度值对应的RGB颜色信息,然后将映射出的颜色信息作为当前灰度图着色后的纹理颜色,见图10。

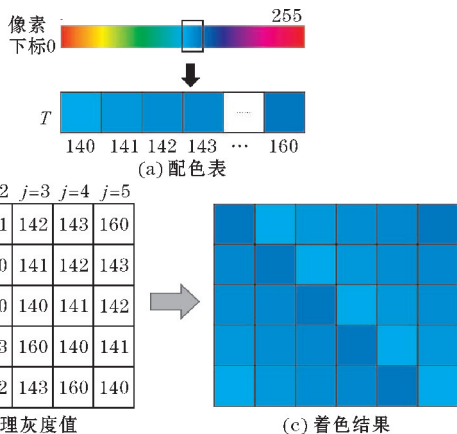


图10 配色表映射过程

使用上述配色表映射方案,将得到的木眼纹理进行着色。结果如图11所示。

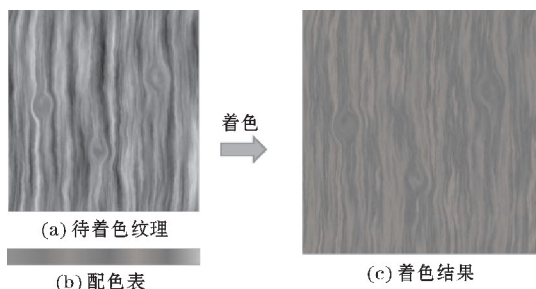


图11 纹理着色结果

1.6 输出贴图

通过上述步骤获取到木制纹理的基础纹理,为获取到更加真实的纹理效果,还需生成AO、HeightMap、Normal、Roughness 4种贴图。其中,获取到的灰度图,可直接作为HeightMap使用。AO、Roughness贴图通过调节灰度图的对比度和亮度得到。亮度的调节方法是直接将灰度的颜色信息加上输入的亮度值即式(17),对比度则是通过式(18)进行调节。

$$\text{color} + = \text{brightness} \quad (17)$$

$$\text{color} = \begin{cases} (\text{color} - 0.5) / (1 - \text{contrast}) + 0.5, & \text{contrast} > 0 \\ (\text{color} - 0.5) / (1 + \text{contrast}) + 0.5, & \text{contrast} \leq 0 \end{cases} \quad (18)$$

通过输入 $\text{brightness} \in [-1, 1]$ 和 $\text{contrast} \in [-1, 1]$,用于控制纹理的亮度和对比度,进而获取纹理的AO和Roughness贴图。

最重要的是法线贴图的生成,本文选择的法线贴图生成算法是根据高度图进行法线的生成,算法思想基于中心差分法。具体做法是:首先,在 u, v 方向上分别计算出灰度值的变化梯度 du 和 dv ;然后,将变化梯度转换到切线空间,得到在切线空间下水平方向的高度变化 $\vec{T}_u = (\text{delta}, 0, du)$ 和垂直方向的高度变化 $\vec{T}_v = (0, \text{setp}, dv)$,其中 delta 是在差分步长 step 下经过式(13)得到的偏移值;最后,再将在切线空间下得到的向量进行叉积运算,将运算结果归一化即可。需要注意的是,叉积的顺序非常重要,这是因为纹理的朝向是 $-z$ 轴,所以会让法线顺着表面所在的朝向,即叉积的顺序是 $\vec{T}_v \times \vec{T}_u$ 。

2 实验结果

实验结果如图12所示,通过基础Albedo图像,生成其余AO、HeightMap、Normal、Roughness贴图,Result则是应用了其余贴图的3D纹理。可以看出,通过本

文的木制纹理合成算法不仅能够模拟出木制纹理的基本特征,同时能够为其添加木眼结构,能够较好地模拟出木制纹理。将纹理贴图应用于模型表面后,呈现出更加真实的纹理,结果见图 13。

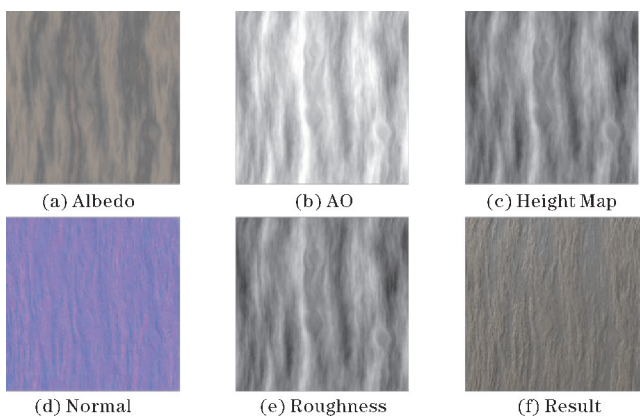


图 12 纹理贴图

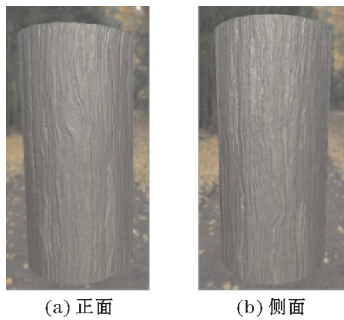


图 13 3D 视图

3 结束语

简单介绍常用的基于人工智能的合成方法、基于样图的纹理合成算法。但基于人工智能的合成方法需要准备大量数据集进行长时间训练,且模型生成纹理质量和数据集的质量相关;基于样图的纹理合成需要提供一张样图,并不能满足在没有样图的情况下合成纹理,同时用户对纹理的可操作空间不够。

针对以上问题,本文使用基于过程纹理合成方法,通过使用 FBM 构建多个不同振幅、频率、比例的柏林噪声,同时对柏林噪声进行缩放、扭曲,以及添加木眼完成对木制纹理的仿真。实验结果表明,该算法能够真实地模拟木制纹理的特征,不仅合成效率十分高效,能够修改纹理相关参数做到实时合成纹理,同时能够多样化合成,纹理更加丰富。

参考文献:

[1] Gatys L, Ecker A S, Bethge M. Texture synthesis u-

sing convolutional neural networks [J]. Advances in neural information processing systems, 2015, 28: 262-270.

[2] Ulyanov D, Lebedev V, Vedaldi A, et al. Texture networks: Feed-forward synthesis of textures and stylized images [C]. International Conference on Machine Learning, 2016: 1349-1357.

[3] Zhou Y, Zhu Z, Bai X, et al. Non-stationary texture synthesis by adversarial expansion [J]. ACM Trans. Graph, 2018, 37(4): 1-13.

[4] 陈凯健, 李二强, 周漾. 基于相对坐标控制的非均匀纹理合成方法 [J]. 计算机辅助设计与图形学学报, 2023, 35(2): 284-292.

[5] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer [C]. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 2001: 341-346.

[6] 徐晓刚, 鲍虎军, 马利庄. 基于相关性原理的多样图纹理合成方法 [J]. 自然科学进展, 2002(6): 107-110.

[7] 靳利贞, 李庆忠. 基于接缝一致性准则的结构纹理图像快速合成算法 [J]. 计算机科学, 2022, 49(6): 262-268.

[8] Peachey D R. Solid texturing of complex surfaces [J]. ACM SIGGRAPH Computer Graphics, 1985, 19(3): 279-286.

[9] Perlin K. An image synthesizer [J]. ACM Siggraph Computer Graphics, 1985, 19(3): 287-296.

[10] Santoni C, Pellacini F. Gtangle: A grammar for the procedural generation of tangle patterns [J]. ACM Transactions on Graphics, 2016, 35(6): 1-11.

[11] Liu J, Gan Y, Dong J, et al. Perception-driven procedural texture generation from examples [J]. Neurocomputing, 2018, 291(5): 21-34.

[12] Larsson M, Ijiri T, Yoshida H, et al. Procedural texturing of solid wood with knots [J]. ACM Transactions on Graphics, 2022, 41(4): 1-10.

[13] 王志强, 张志伟, 杨海泉. 基于 Perlin 噪声的花卉仿真方法 [J]. 深圳大学学报(理工版), 2019, 36(4): 460-466.

[14] 陈国栋, 苏志鹏, 李剑斌. 基于柏林噪声的心脏纹理图细化 [J]. 佳木斯大学学报(自然科学版), 2018, 36(6): 898-901.

[15] 霍星, 檀结庆. 噪声函数在木纹纹理中的应用 [J]. 合肥工业大学学报(自然科学版), 2005

(11):1465–1467.

on Graphics,2002,21(3):681–682.

[16] Perlin K. Improvingnoise[J]. ACM Transactions

A Procedural Texture Synthesis Method based on Perlin Noise and Fractal Brownian Motion

LI Jiaru, HE Xiaoxi, LIU Yinghu, MENG Fanlin, ZHU Qun

(College of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: To solve the problems of uncontrollable, inflexible and low real-time synthesis process in the existing artificial intelligence texture generation, and sample based texture synthesis, this paper uses procedural texture generation technology to processes a texture generation algorithm based on Perlin noise and fractal Brownian motion, which is used to synthesize the features of real wood texture. Firstly, the texture space to be generated is divided into regular and uniform grid points, and a gradient vector is randomly generated in each grid point. Then, the gradient vector on the grid points is interpolated using the tri-linear interpolation method. At the same time, the noise patterns with multiple frequencies and amplitudes are weighted and superposed by the fractal algorithm to generate a smooth texture pattern, and on this basis, the texture is scaled, distorted Add wooden eyes to simulate realistic texture structures. Experiments have shown that this algorithm can truly simulate the characteristics of wooden textures, and this synthesis method has advantages such as real-time, efficient, and strong customizability, which has broad application value in fields such as games, modeling, and virtual reality.

Keywords: procedural texture; Perlin noise; fractal; interpolation; real time