浙江科技学院学报,第 29 卷第 3 期,2017 年 6 月
Journal of Zhejiang University of Science and Technology
Vol. 29 No. 3, Jun. 2017

doi: 10.3969/j.issn.1671-8798.2017.03.009

实时交互式三维模型纹理映射算法

胡中天,叶　绿
(浙江科技学院 信息与电子工程学院,杭州 310023)

摘　要:针对三维动画制作中模型纹理绘制不够直观、效率低下等问题,提出一种实时交互式三维模型纹理映射算法,即通过交互式三维拾取准确找到三维模型与二维纹理图像对应点,直接在三维模型表面使用笔刷进行纹理图像的绘制。实验在 Unity 3D 引擎中进行,结果表明用户可以使用该算法进行纹理的交互式设计与合成,其笔刷定位精确,绘制过程帧率稳定,绘制得到的纹理图像质量较高。该方法对比传统制作流程中对三维模型展开 UV,在二维图像处理软件中进行纹理绘制的方法更为直观、高效。

关键词:纹理映射;三维重建;二维图像;交互
中图分类号:TP391.41　　**文献标志码**:A　　**文章编号**:1671-8798(2017)03-0206-08

A realtime interactive 3D model texture mapping algorithm

HU Zhongtian, YE Lü
(School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, Zhejiang, China)

Abstract: In response to such problems as poorly visualized and inefficient drawing of model texture in the process of 3D animation production, the article puts forward a real-time interactive 3D model texture mapping algorithm. The algorithm can accurately locate points of 3D models corresponding to 2D texture images through interactive 3D picking and can directly use a brush to draw texture images on the surface of 3D models. An experiment conducted in the Unity 3D engine shows that users can apply the algorithm to interactive design and composition, and achieve high-quality texture images, thanks to accurate localization of the brush and stable frame rate in the drawing process. Compared with the traditional process in which 3D models are unfolded, the algorithm proves more visualized and efficient in terms of ways of drawing texture in 2D image processing software.

Keywords: texture mapping; 3D reconstruction; 2D image; interaction

收稿日期:2017-03-07
基金项目:浙江科技学院大学生课外科技创新与实践项目(春萌计划)(浙科院团[2016]2 号)

通信作者:叶　绿(1962—　),女,浙江省杭州人,教授,博士,主要从事图形学、图像处理、三维重建和虚拟现实研究。E-mail: yelue@zust.edu.cn。

纹理映射是计算机图形学中常用的一种增强虚拟物体真实感的技术。随着近年来三维技术的发展，三维动画制作、虚拟现实、工业仿真等行业中都对虚拟物体的真实感与精度提出了更高的要求，纹理映射也被广泛应用到各种实际项目中。

1974 年 Catmull[1-3] 首次提出了纹理映射的概念，通过纹理空间内的 UV 坐标与三维物体表面局部坐标之间的映射关系，将纹理图像中的彩色参数值映射到对应位置的三维曲面上，使得三维物体表面能够呈现纹理图像中的色彩效果。1986 年 Bier 等提出了两步映射法[4]，三维物体表面的彩色参数值由中介曲面二次映射得到，这种纹理图像先被映射到中介曲面的方法，提高了纹理映射的效率，但应用于复杂结构的物体表面时可能出现破洞、接缝等问题。1999 年 Niem[5] 提出利用多相机多角度拍摄不同焦距的图像素材，再将纹理图像映射于重新剖分三角面的物体模型表面的映射方法，该方法映射效果较好，但算法比较复杂，不易实现。文献[6]中提出一种基于物体单色几何模型的纹理映射算法，以三角面片作为中介面，能够将照片序列上的纹理图像映射到圆柱模型上，并通过拼接、变形等过程获得物体的纹理；该方法不需要质量很高的原始纹理，允许在运行时进行交互操作，但对于复杂模型表面可能存在部分位置的纹理无法映射的情况，且计算量较大。文献[7]中提出基于 Catmull 映射方法的一种交互式纹理映射方法，该方法通过三维拾取选择纹理坐标与三维曲面的对应特征点进行纹理映射，只通过鼠标选取一些特征点后将一张纹理贴图整体映射到模型表面，其交互操作仍有一定的局限性。

在上述的映射方法中，映射得到的结果纹理中无法保证纹理贴图的完整性，如出现缺口、接缝等。文献[8]提出利用梯度融合方法以消除纹理拼接边界的颜色差异并消除贴图接缝。文献[9]提出一种多参数加权的纹理映射算法以消除纹理接缝，对网格结构误差较大的区域也能取得良好效果，并可应用于城市级别的大场景三维重建。但是，修复算法的计算量都较大，难以实现实时交互。

笔者在分析、总结前人研究的基础上，提出一种基于离散面片的实时交互式三维模型纹理映射算法。该算法使用户能够通过笔刷实时绘制方式进行贴图的映射，其算法实现简单，计算量小，映射结果质量较高，不存在贴图接缝问题。

1　三维图形渲染管线与坐标位置映射

1.1　计算机三维图形渲染管线

图 1 为三维图形渲染管线，也称为渲染流水线，是显示芯片内部处理图形信号相互独立的并行处理单元。渲染管线主要分为应用程序阶段、几何阶段、光栅阶段 3 个阶段。应用程序阶段将计算好的顶点坐标、法向量、纹理坐标、纹理数据通过数据总线传给图形硬件，作为进一步处理的源数据。几何阶段主要负责顶点坐标变换、光照、裁剪、投影及屏幕映射。光栅阶段则是将像素的颜色值写入帧缓冲。

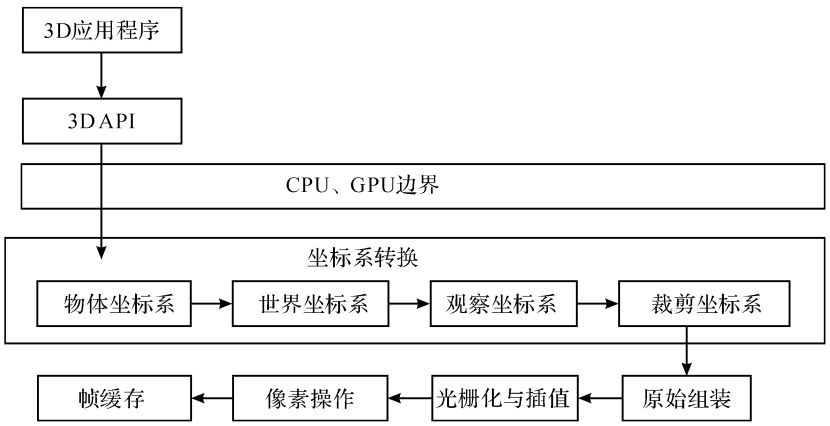


图 1　三维图形渲染管线

Fig. 1　3D graphics rendering pipeline

1.2 三维物体表面与二维纹理空间的位置映射

渲染管线的光栅化阶段中包括一步纹理操作，根据像素的纹理坐标查询对应的纹理值。具体而言，如对一个三维空间下的立方体进行展开(图 2)，将它的 6 个面都展开到同一个平面后就可以方便地对每个表面进行颜色纹理的绘制。这个过程就是三维模型的 UV 展开，三维物体表面与二维纹理空间的位置映射实质上就是这样找到模型表面位置和二维纹理坐标对应关系的过程。

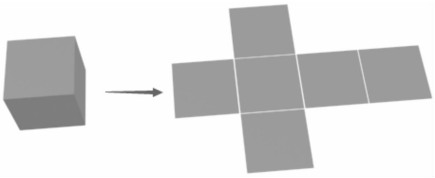


图2 立方体 UV 展开

Fig. 2 Cube's UV unfolding

2 三维模型纹理映射的新方法

2.1 交互式三维拾取

渲染管线的几何阶段中，根据顶点坐标变换的先后顺序，主要有模型坐标空间、世界坐标空间、观察坐标空间和屏幕坐标空间。模型坐标空间中记录模型顶点的坐标值，这些值是在建立模型时得到的；世界坐标空间中记录世界坐标原点，物体的世界空间坐标通过物体自身位置与原点位置相比较得到；观察坐标空间则是以摄像机为原点，由视线方向、视角和远近平面组成的一个视锥空间；屏幕坐标空间是一个单位立方体空间，这个立方体被称为规范立方体(CCV)，CCV 的近平面即对应最终用户所看到的二维屏幕空间。

交互式三维拾取操作是指当用户在屏幕上用鼠标点击某个图元时，应用程序能够返回该图元的某些相关信息。拾取过程中，首先获取用户在屏幕空间下点击的坐标位置，转换为观察坐标空间下的坐标位置，接着从观察坐标空间转换到世界坐标空间，获取处于当前世界坐标空间目标点的三维物体，最终将目标点坐标转换到模型坐标空间下，再通过三维物体表面与二维纹理空间的位置映射关系，得到对应的二维纹理坐标。

对于模型上即空间中一点的世界坐标系为$[X_w,Y_w,Z_w]^T$，转换到相机坐标系时增加一维用齐次坐标来表示它，以方便对它进行平移操作：

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \tag{1}$$

式(1)中：$[X_c,Y_c,Z_c]^T$ 为相机坐标；$[X_w,Y_w,Z_w,1]^T$ 为模型所在的世界坐标；$\boldsymbol{R}$ 为旋转矩阵；$\boldsymbol{T}$ 为平移矩阵。

忽略畸变过程，相机坐标系通过焦距对角矩阵和畸变系数转换到图像物理坐标系：

$$Z_c \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \tag{2}$$

式(2)中：f 为焦距；$[X_c,Y_c,Z_c]^T$ 为相机坐标。

图像物理坐标系通过像素转换矩阵转换到像素坐标系：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\mathrm{d}x} & 0 & u_0 \\ 0 & \frac{1}{\mathrm{d}y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{3}$$

式(3)中：$\mathrm{d}x$ 和 $\mathrm{d}y$ 分别为 x 方向和 y 方向的一个像素所占的长度单位；u_0,v_0 表示图像的中心像素坐标和图像原点像素坐标之间相差的横向和纵向像素数；$[X,Y,1]^T$ 为归一化后的图像物理坐标。

故从鼠标点击拾取的屏幕像素坐标点(u,v)转换到物体被拾取点所在的世界坐标(X_w,Y_w,Z_w)的转换过程总公式如下：

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = Z_c \boldsymbol{R}^{-1} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \boldsymbol{R}^{-1}\boldsymbol{T}。$$

2.2　离散化模型表面上色

2.2.1　精确笔刷定位

笔刷作为单独的一个三维物体，以平面方式增加透明贴图渲染，通过鼠标在屏幕坐标空间下的位置获取模型坐标空间位置后，同时获取模型表面该坐标位置的法线向量，通过这两个参数即可确定当前位移的笔刷位置及其朝向。

2.2.2　笔刷轨迹的离散化

笔刷轨迹的离散化是将连续的笔刷移动轨迹离散为不连续点迹的过程。在用户移动鼠标的过程中，笔刷跟随鼠标紧贴模型表面移动，每间隔一小段时间对笔刷轨迹进行采样，记录下当前笔刷的位置坐标。

2.2.3　根据离散点迹对模型表面上色

如图 3 所示，每进行一次采样，在采样位置实例化一个紧贴模型表面的带颜色三维面片，当采样率足够高时，三维面片即可组成连续的线段。三维面片依照产生的次序进行命名，可以依据三维面片的名称逐个依次删除来撤销操作，达到类似平面图像处理软件中历史记录的作用。

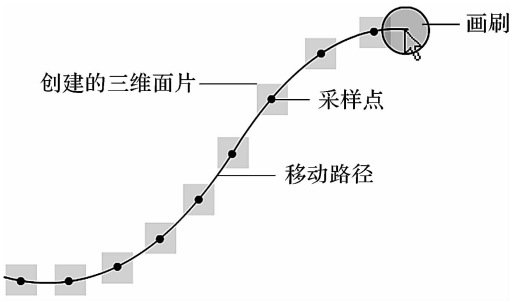


图 3　路径采样与三维面片创建

Fig. 3　Path sampling and 3D patch creation

2.2.4　转换模型表面绘画信息到纹理贴图

模型表面的绘画信息由所有三维面片组成，转换时，对模型表面的颜色进行逐像素采样，将获取到的颜色值绘制到纹理贴图的对应位置，更新模型的纹理贴图，并删除所有三维面片。

2.3　算法主要流程及关键代码

图 4 为实时交互式三维模型纹理映射算法的主要算法流程，算法关键部分伪代码：

1）获取用户当前鼠标在屏幕空间的坐标位置并转换到模型 UV 坐标。

```
mouseInputVector←用户当前鼠标在屏幕空间的坐标位置；
uvRay←从 mouseInputVector 向观察方向发射一条射线；
if uvRay 射线检测返回了碰撞物体 THEN BEGIN
    meshCollider←返回的碰撞物体；
    collidPosition←计算 uvRay 与 meshCollider 碰撞点的世界坐标；
    uvPosition←collidPosition 转换为对应的纹理坐标位置(UV 坐标)；
    return true；
END
return false；
```

2）对采样点创建三维面片。

```
if 获取到 UV 坐标且处于绘制状态 THEN BEGIN
    temp_brush←创建面片；
    temp_brush. worldPosition←画刷位置；
    temp_brush. localPosition←uvPosition；
    temp_brush. localScale←画刷大小；
    temp_brush. name←面片实例编号名称；
```

添加新产生的面片到面片列表；

面片计数值加一；

END

3)保存当前绘制的纹理贴图。

tex←创建新的 2D 纹理；

逐像素读取颜色到 tex；

保存纹理贴图为选定格式；

删除列表中的面片；

清除场景中的面片实例；

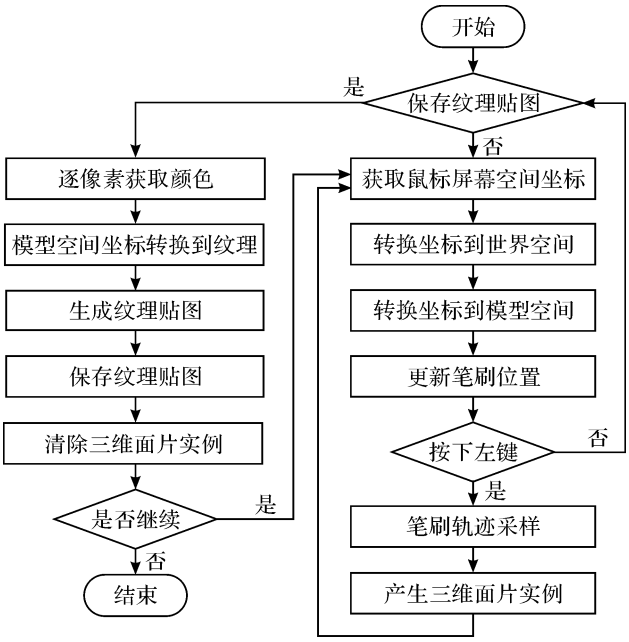


图4 实时交互式三维模型纹理映射算法流程

Fig. 4 Process of the algorithm for realtime interactive 3D model texture mapping

3 实验结果与分析

系统运行在一台 Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 四核 CPU，NVIDIA GeForce 820M 显卡的计算机上，Unity 3D 版本为 5.3.4f1(64 bit)，渲染时均使用标准着色器，测试纹理分辨率为 2 048×2 048 像素大小。

表 1 所示为交互式三维拾取的部分坐标转换结果。

表 1 交互式三维拾取坐标转换结果

Table 1 Coordinate conversion results of interactive 3D picking

鼠标屏幕像素坐标		射线检测碰撞点坐标			UV 坐标(world)			UV 坐标(local)	
x	y	x	y	z	u	v	w	u	v
369.000	256.000	−0.423	0.323	−0.500	0.423	−0.323	1.000	0.923	0.177
582.000	295.000	0.129	0.287	−0.384	−0.199	0.183	1.000	0.301	0.683
466.000	281.000	−0.079	0.242	−0.426	−0.079	0.242	1.000	0.220	0.654
614.000	46.000	0.167	−0.152	−0.440	−0.194	−0.096	1.000	0.306	0.404

图 5 为不同采样率下进行绘画能够获得的最终效果，采样率越高，绘画越接近连续的线段。图 6 是在不同形状模型表面的绘制效果。图 7(a)是通过添加遮罩改变笔刷形状后绘制的效果，图7(b)是对三

维面片本身增加图片作为纹理后，用于绘制组成模型整体纹理的效果，可以产生比纯色笔刷绘画更丰富的色彩效果。

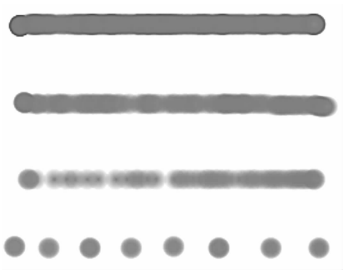
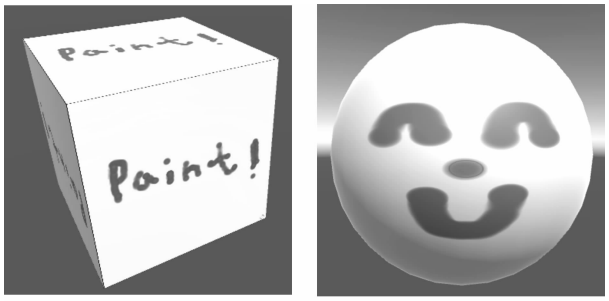


图 5　不同采样率下的画线效果

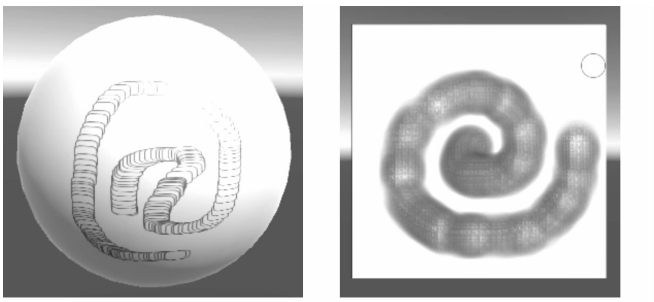
Fig. 5　Effect of lines drawn at different sampling rates



(a) 立方体表面　　(b) 球体表面

图 6　不同形状表面绘制效果

Fig. 6　Effects of drawing on different shape surface

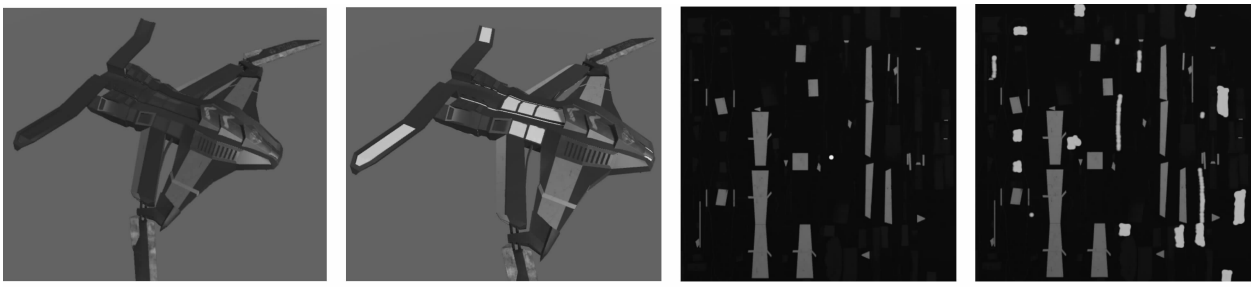


(a) 通过遮罩改变画刷形状　　(b) 彩色图片作为三维面片纹理

图 7　增加画刷遮罩及图片作为三维面片纹理的绘制效果

Fig. 7　Effects of drawing with brush cover added and picture as 3D patch's texture

图 8 是在模型原有纹理的基础上进行混合绘制的效果，可以快速高效地增加模型的细节且不影响最终贴图质量。



(a) 使用原贴图模型效果　　(b) 使用混合后贴图模型效果　　(c) 原贴图效果　　(d) 混合后效果

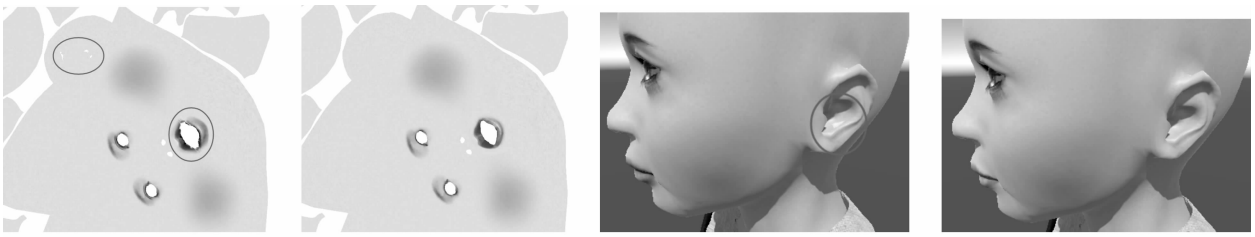
图 8　混合纹理效果

Fig. 8　Effect of mixed texture

图 9 为用于混合映射的原始照片素材，使用不同方法映射于模型表面生成纹理贴图(图 10)。图 10(a)与(c)为使用两步映射法的贴图混合映射生成的纹理贴图效果与模型效果，在复杂结构处以及嘴部内边缘处产生了破洞与明显接缝；而图 10(b)与(d)中，采用本文先创建离散化面片对模型表面上色，再进行重采样创建纹理贴图的方法，就不存在上述问题。对比文献[8]中利用梯度融合方法消除纹理拼接边界的颜色差异的方法与文献[9]中多参数加权的纹理映射算法消除纹理接缝的方法，本文方法不需要额外消耗大量时间进行边界处理。

图 9 用于混合映射的原始纹理素材

Fig. 9 Original texture assets for mixed mapping



(a) 直接贴图混合映射效果 (b) 本文方法映射效果 (c) 直接贴图混合映射模型效果 (d) 本文方法映射模型效果

图 10 边缘效果对比

Fig. 10 Comparison of marginal effects

图 11 为实验系统运行时的实时系统资源统计，方框中 CPU 与内存占用峰值时段为纹理绘制时段。在模型表面进行贴图绘制时，应用程序运行帧率始终保持 60 帧/s 以上，最高达到 200 帧/s，不会产生卡顿且贴图像素大小可调，而且保存后的纹理贴图效果与实时绘制时效果一致，无偏差。

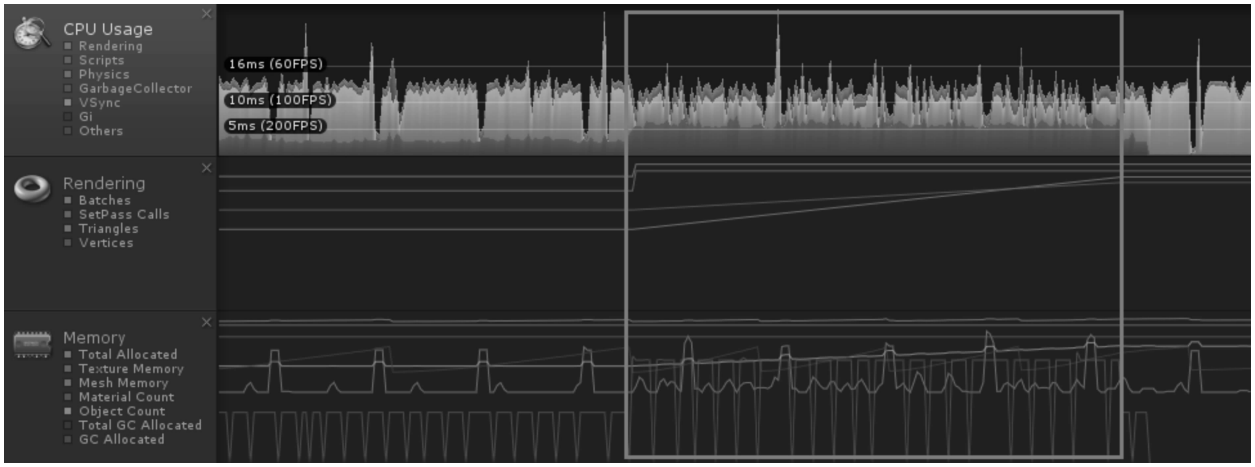


图 11 系统资源统计

Fig. 11 Statistics of profiler

对比文献[5]中的方法，本文算法无需将被映射物体重新分割为三角面，对比文献[8-9]，本文算法无需实现映射纹理的错误边缘与破洞修复，算法实现简单。与文献[10-13]中的纹理映射方法作对比，本文算法不仅能够将二维图片直接映射到三维物体表面作为纹理贴图，还可以让用户使用笔刷交互式地进行纹理的绘画，笔刷支持纯色也支持使用图片作为三维面片的纹理进行模型的纹理绘制，同时支持大小调节与基于遮罩的形状调节，从而保证了用户最大的创作自由度。比起在平面图像处理软件中进行贴图绘制的方式，本文方法的绘制方式更为直观高效，所见即所得。

4 结　论

笔者提出一种实时交互式三维模型纹理映射算法，通过交互式三维拾取方式直接在三维模型表面使用笔刷进行纹理图像的绘制。笔刷支持纯色也支持使用图片作为三维面片的纹理进行模型的纹理绘制，同时支持大小调节与基于遮罩的形状调节，还支持对历史记录进行撤销。对比在平面图像处理软件中进行贴图绘制的方式，本文方法的绘制方式更为直观高效，对提高三维模型贴图阶段制作效率、提升成品质量有一定作用。但本文算法目前还不支持类似 PhotoShop 等传统平面图像处理软件中色阶、色调、曝光度、对比度调整等更复杂的图像处理功能，这是今后需要改进的方面。

参考文献：

[1] CATMULL E. A subdivision algorithm for computer display of curved surfaces[R]. Salt Lake City: The University of Utah, 1974.

[2] CATMULL E. A hidden-surface algorithm with anti-aliasing[J]. ACM Siggraph Computer Graphics, 1978, 12(3): 6.

[3] CATMULL E, SMITH A R. 3-D transformations of images in scanline order[J]. ACM Siggraph Computer Graphics, 1980, 14(3): 279.

[4] BIER E A, SLOAN K R. Two-part texture mappings[J]. IEEE Computer Graphics and Applications, 1986, 6(9): 40.

[5] NIEM W, WINGBERMUHLE J. Automatic reconstruction of 3D objects using a mobile monoscopic camera[J]. Image and Vision Computing, 1999, 17(2): 125.

[6] 陈任，鲁东明，潘云鹤. 基于几何模型与照片序列的不规则物体纹理获取[J]. 中国图象图形学报，2003，8(8)：902.

[7] 王琰，王明宇. 一种交互式纹理映射方法[J]. 小型微型计算机系统，2011，32(10)：2101.

[8] 姜翰青，王博胜，章国锋，等. 面向复杂三维场景的高质量纹理映射[J]. 计算机学报，2015，38(12)：2349.

[9] 刘彬，陈向宁，薛俊诗. 多参数加权的无缝纹理映射算法[J]. 中国图象图形学报，2015，20(7)：929.

[10] 姚砺，钱朔. 基于三角网格模型的局部纹理映射[J]. 计算机应用与软件，2015，32(3)：205.

[11] 万燕，王慧洁，鲁俊. 基于三角网格模型的纹理映射研究[J]. 计算机应用与软件，2016，33(4)：160.

[12] 唐晶磊，景旭，何东健. 基于两步法的交互式纹理映射研究[J]. 计算机工程与设计，2008，29(8)：2062.

[13] 陈益，王莉莉. 基于几何网格的交互式纹理合成方法[J]. 计算机辅助设计与图形学学报，2013，25(10)：1480.