



(12)发明专利

(10)授权公告号 CN 107463398 B

(45)授权公告日 2018.08.17

(21)申请号 201710600237.7

G06T 15/00(2011.01)

(22)申请日 2017.07.21

(56)对比文件

(65)同一申请的已公布的文献号

申请公布号 CN 107463398 A

(43)申请公布日 2017.12.12

CN 101364310 A, 2009.02.11,

CN 101840566 A, 2010.09.22,

CN 105931180 A, 2016.09.07,

CN 105957133 A, 2016.09.21,

US 6587104 B1, 2003.07.01,

(73)专利权人 腾讯科技(深圳)有限公司

刘浩.多尺度三维灾情场景构建与动态标绘  
关键技术研究.《中国博士学位论文全文数据库  
基础科学辑》.2013,

地址 518057 广东省深圳市南山区高新区  
科技中一路腾讯大厦35层

审查员 唐丹颖

(72)发明人 徐敏君

(74)专利代理机构 广州三环专利商标代理有限  
公司 44202

代理人 郝传鑫 熊永强

(51)Int.Cl.

G06F 9/445(2018.01)

G06T 11/00(2006.01)

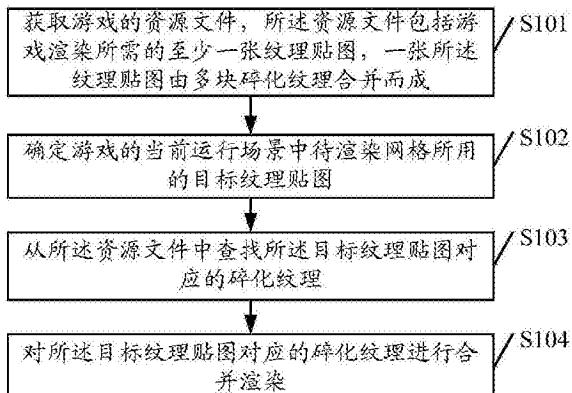
权利要求书3页 说明书15页 附图5页

(54)发明名称

游戏渲染方法、装置、存储设备及终端

(57)摘要

本发明实施例公开一种游戏渲染方法、装置、存储设备及终端，在资源打包过程中即将纹理贴图碎化处理为多块碎化纹理，按照游戏渲染需要加载所用的目标纹理贴图对应的碎化纹理，能够有效的降低内存加载开销，减小内存负担；另外，对目标纹理贴图对应的碎化纹理进行合并渲染，能够保证渲染画面效果，避免内存负担随游戏运行线性增加，从而保证游戏运行顺畅。



1. 一种游戏渲染方法,其特征在于,包括:

获取游戏渲染所需的至少一张纹理贴图,所述纹理贴图包含原始纹理数据;

依次选取各张纹理贴图,将所选纹理贴图按照第一预设规格进行切割得到多个矩形框;

对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多个碎化纹理数据,并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据;

将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理,一块碎化纹理包括一个碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据;

将各纹理贴图对应的碎化纹理封装至资源文件中;

获取游戏的资源文件,所述资源文件包括游戏渲染所需的至少一张纹理贴图,其中,一张所述纹理贴图由多块碎化纹理合并而成;

确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图;

从所述资源文件中查找所述目标纹理贴图对应的碎化纹理;

对所述目标纹理贴图对应的碎化纹理进行合并渲染。

2. 如权利要求1所述的游戏渲染方法,其特征在于,第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和。

3. 如权利要求2所述的游戏渲染方法,其特征在于,所述将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理,包括:

在每个矩形框上下两侧各预留预设宽度的边缘,并在每个矩形框左右两侧各增加预设宽度的边缘,使每个矩形框具备中央区和包围中央区的边缘区;

将多个碎化纹理数据分别填充至对应矩形框的中央区,并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区,一个完成填充的矩形框形成一块碎化纹理。

4. 如权利要求1所述的游戏渲染方法,其特征在于,所述将各纹理贴图对应的碎化纹理封装至资源文件中,包括:

获取各纹理贴图的属性,所述属性包括ID、释放类型及偏移信息,所述释放类型包括立即释放、自动释放或常驻内存;

根据各纹理贴图的属性生成简要信息表;

将所述简要信息表存放至资源文件的文件头,将各纹理贴图对应的碎化纹理存放至资源文件的文件主体。

5. 如权利要求2-4任一项所述的游戏渲染方法,其特征在于,所述对所述目标纹理贴图对应的碎化纹理进行合并渲染,包括:

按照第一预设规格将内存中的预置动态加载区进行切割,形成多个动态加载行;

采用装箱算法对所述动态加载行的空闲区域进行分配;

依次将所述目标纹理贴图对应的碎化纹理加载至所述动态加载行的空闲区域;

从所述动态加载行中读取所加载的碎化纹理并提交至游戏的渲染引擎进行渲染。

6. 如权利要求4所述的游戏渲染方法,其特征在于,所述对所述目标纹理贴图对应的碎

化纹理进行合并渲染之后,还包括:

从资源文件的文件头中获取所述目标纹理贴图的释放类型;

按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行;将回收的动态加载行进行合并。

7. 一种游戏渲染装置,其特征在于,包括:

资源打包处理单元,用于获取游戏渲染所需的至少一张纹理贴图,所述纹理贴图包含原始纹理数据,分别将各纹理贴图碎化处理为多块碎化纹理,一块碎化纹理包括一个碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据,以及将各纹理贴图对应的多块碎化纹理封装至资源文件中;

文件获取单元,用于获取游戏的资源文件,所述资源文件包括游戏渲染所需的至少一张纹理贴图,其中,一张所述纹理贴图由多块碎化纹理合并而成;

目标确定单元,用于确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图;

查找单元,用于从所述资源文件中查找所述目标纹理贴图对应的碎化纹理;

合并渲染单元,用于对所述目标纹理贴图对应的碎化纹理进行合并渲染;

其中,所述资源打包处理单元具体用于:依次选取各张纹理贴图,将所选纹理贴图按照第一预设规格进行切割得到多个矩形框;对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多个碎化纹理数据,并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据;将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理。

8. 如权利要求7所述的游戏渲染装置,其特征在于,所述游戏渲染装置还包括:

回收处理单元,用于从资源文件的文件头中获取所述目标纹理贴图的释放类型,按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行,以及将回收的动态加载行进行合并。

9. 如权利要求8所述的游戏渲染装置,其特征在于,第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和;

其中,将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理包括:在每个矩形框上下两侧各预留预设宽度的边缘,并在每个矩形框左右两侧各增加预设宽度的边缘,使每个矩形框具备中央区和包围中央区的边缘区;将多个碎化纹理数据分别填充至对应矩形框的中央区,并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区,一个完成填充的矩形框形成一块碎化纹理;以及,

所述资源打包处理单元具体用于:获取各纹理贴图的属性,所述属性包括ID、释放类型及偏移信息,所述释放类型包括立即释放、自动释放或常驻内,根据各纹理贴图的属性生成简要信息表,以及将所述简要信息表存放至资源文件的文件头,将各纹理贴图对应的碎化纹理存放至资源文件的文件主体。

10. 如权利要求8或9所述的游戏渲染装置,其特征在于,所述合并渲染单元具体用于:按照第一预设规格将内存中的预置动态加载区进行切割,形成多个动态加载行;采用装箱算法对所述动态加载行的空闲区域进行分配;依次将所述目标纹理贴图对应的碎化纹理加载至所述动态加载行的空闲区域;以及,从所述动态加载行中读取所加载的碎化纹理并

提交至游戏的渲染引擎进行渲染。

11. 一种存储设备,其特征在于,所述存储设备存储有一条或一条以上指令,所述一条或一条以上指令适于由处理器加载并执行如权利要求1-6任一项所述的游戏渲染方法。

12. 一种游戏渲染终端,其特征在于,包括:

处理器,适于实现一条或一条以上指令;以及,

存储设备,所述存储设备存储有一条或一条以上指令,所述一条或一条以上指令适于由所述处理器加载并执行如权利要求1-6任一项所述的游戏渲染方法。

## 游戏渲染方法、装置、存储设备及终端

### 技术领域

[0001] 本发明涉及互联网技术领域,具体涉及互联网游戏技术领域,尤其涉及一种游戏渲染方法、一种游戏渲染装置、一种存储设备及一种游戏渲染终端。

### 背景技术

[0002] 传统的游戏渲染方法,一般是预先将游戏的当前运行场景中可能需要使用的所有纹理贴图(如景物、角色和UI(User Interface,用户界面)等)全部加载至内存中,当需要对某一纹理贴图进行渲染时,从内存中直接获取该纹理贴图,然而很多纹理贴图其实并不是一直在使用,将其加载至内存中,既增加了加载开销,也增加了内存占用,进而造成游戏卡顿。现有技术中还出现另一种游戏渲染方法,能够一定程度上减小内存负担,该方法在资源打包过程中使用纹理合并工具texture packer把多张小纹理贴图合并到一张大纹理贴图中,在需要对多个游戏元素(如景物、角色等)进行渲染时才将该大纹理贴图加载至内存中,从而节省一定加载开销;在渲染多个游戏元素时只要这些元素使用的是同一张大纹理贴图,并且渲染参数一致便可以合并渲染批次,从而提升渲染性能。这种使用纹理合并工具的方法虽然可以提高渲染性能,但需要整张大纹理贴图同时加载到内存,由于游戏运行时存在着大量的纹理复用,这样的方式同样会造成大量的不必要的内存开销,并且随着游戏版本更新和内容增加而使得内存负担显著上升,进而造成游戏卡顿。

### 发明内容

[0003] 本发明实施例提供一种游戏渲染方法、装置、存储设备及终端,能够保证渲染画面效果,避免内存负担随游戏运行线性增加,从而保证游戏运行顺畅。

[0004] 一方面,本发明实施例提供一种游戏渲染方法,可包括:

[0005] 获取游戏的资源文件,所述资源文件包括游戏渲染所需的至少一张纹理贴图,其中,一张所述纹理贴图由多块碎化纹理合并而成;

[0006] 确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图;

[0007] 从所述资源文件中查找所述目标纹理贴图对应的碎化纹理;

[0008] 对所述目标纹理贴图对应的碎化纹理进行合并渲染。

[0009] 优选地,所述获取游戏的资源文件之前,还包括:

[0010] 获取游戏渲染所需的至少一张纹理贴图,所述纹理贴图包含原始纹理数据;

[0011] 分别将各纹理贴图碎化处理为多块碎化纹理,一块碎化纹理包括一个碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据;

[0012] 将各纹理贴图对应的碎化纹理封装至资源文件中。

[0013] 优选地,所述分别将各纹理贴图碎化处理为多块碎化纹理,包括:

[0014] 依次选取各张纹理贴图,将所选纹理贴图按照第一预设规格进行切割得到多个矩形框;

[0015] 对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多

个碎化纹理数据，并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据；

[0016] 将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框，一个完成填充的矩形框形成一块碎化纹理；

[0017] 其中，第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和。

[0018] 优选地，所述将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框，一个完成填充的矩形框形成一块碎化纹理，包括：

[0019] 在每个矩形框上下两侧各预留预设宽度的边缘，并在每个矩形框左右两侧各增加预设宽度的边缘，使每个矩形框具备中央区和包围中央区的边缘区；

[0020] 将多个碎化纹理数据分别填充至对应矩形框的中央区，并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区，一个完成填充的矩形框形成一块碎化纹理。

[0021] 优选地，所述将各纹理贴图对应的碎化纹理封装至资源文件中，包括：

[0022] 获取各纹理贴图的属性，所述属性包括ID(Identity, 身份标识)、释放类型及偏移信息，所述释放类型包括立即释放、自动释放或常驻内存；

[0023] 根据各纹理贴图的属性生成简要信息表；

[0024] 将所述简要信息表存放至资源文件的文件头，将各纹理贴图对应的碎化纹理存放至资源文件的文件主体。

[0025] 优选地，所述对所述目标纹理贴图对应的碎化纹理进行合并渲染，包括：

[0026] 按照第一预设规格将内存中的预置动态加载区进行切割，形成多个动态加载行；

[0027] 采用装箱算法对所述动态加载行的空闲区域进行分配；

[0028] 依次将所述目标纹理贴图对应的碎化纹理加载至所述动态加载行的空闲区域；

[0029] 从所述动态加载行中读取所加载的碎化纹理并提交至游戏的渲染引擎进行渲染。

[0030] 优选地，所述对所述目标纹理贴图对应的碎化纹理进行合并渲染之后，还包括：

[0031] 从资源文件的文件头中获取所述目标纹理贴图的释放类型；

[0032] 按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行；

[0033] 将回收的动态加载行进行合并。

[0034] 另一方面，本发明实施例提供一种游戏渲染装置，可包括：

[0035] 文件获取单元，用于获取游戏的资源文件，所述资源文件包括游戏渲染所需的至少一张纹理贴图，其中，一张所述纹理贴图由多块碎化纹理合并而成；

[0036] 目标确定单元，用于确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图；

[0037] 查找单元，用于从所述资源文件中查找所述目标纹理贴图对应的碎化纹理；

[0038] 合并渲染单元，用于对所述目标纹理贴图对应的碎化纹理进行合并渲染。

[0039] 优选地，所述游戏渲染装置还包括：

[0040] 资源打包处理单元，用于获取游戏渲染所需的至少一张纹理贴图，所述纹理贴图包含原始纹理数据，分别将各纹理贴图碎化处理为多块碎化纹理，一块碎化纹理包括一个

碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据,以及将各纹理贴图对应的多块碎化纹理封装至资源文件中;以及,

[0041] 回收处理单元,用于从资源文件的文件头中获取所述目标纹理贴图的释放类型,按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行,以及将回收的动态加载行进行合并。

[0042] 优选地,所述资源打包处理单元具体用于:依次选取各张纹理贴图,将所选纹理贴图按照第一预设规格进行切割得到多个矩形框;对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多个碎化纹理数据,并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据;将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理;其中,第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和;

[0043] 其中,将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理包括:在每个矩形框上下两侧各预留预设宽度的边缘,并在每个矩形框左右两侧各增加预设宽度的边缘,使每个矩形框具备中央区和包围中央区的边缘区;将多个碎化纹理数据分别填充至对应矩形框的中央区,并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区,一个完成填充的矩形框形成一块碎化纹理;以及,

[0044] 所述资源打包处理单元具体用于:获取各纹理贴图的属性,所述属性包括ID、释放类型及偏移信息,所述释放类型包括立即释放、自动释放或常驻内,根据各纹理贴图的属性生成简要信息表,以及将所述简要信息表存放至资源文件的文件头,将各纹理贴图对应的多块碎化纹理存放至资源文件的文件主体。

[0045] 优选地,所述合并渲染单元具体用于:按照第一预设规格将内存中的预置动态加载区进行切割,形成多个动态加载行;采用装箱算法对所述动态加载行的空闲区域进行分配;依次将所述目标纹理贴图对应的各块碎化纹理加载至所述动态加载行的空闲区域;以及,从所述动态加载行中读取所加载的碎化纹理并提交至游戏的渲染引擎进行渲染。

[0046] 再一方面,本发明实施例还提供一种存储设备,所述存储设备存储有一条或一条以上指令,所述一条或一条以上指令适于由处理器加载并执行本发明实施例所述的游戏渲染方法。

[0047] 再一方面,本发明实施例还提供一种游戏渲染终端,所述游戏渲染终端包括:

[0048] 处理器,适于实现一条或一条以上指令;以及,

[0049] 存储设备,所述存储设备存储有一条或一条以上指令,所述一条或一条以上指令适于由所述处理器加载并执行本发明实施例所述的游戏渲染方法。

[0050] 本发明实施例在资源打包过程中将纹理贴图碎化处理为多块碎化纹理,按照游戏渲染需要加载所用的目标纹理贴图对应的碎化纹理,能够有效的降低内存加载开销,减小内存负担;另外,对目标纹理贴图对应的碎化纹理进行合并渲染,能够保证渲染画面效果,避免内存负担随游戏运行线性增加,从而保证游戏运行顺畅。

## 附图说明

[0051] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现

有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

- [0052] 图1为本发明实施例提供的一种游戏渲染方法的流程图;
- [0053] 图2为本发明实施例提供的另一种游戏渲染方法的流程图;
- [0054] 图3a-图3c为本发明实施例提供的碎化处理过程的示意图;
- [0055] 图4为本发明实施例提供的合并渲染的画面效果示意图;
- [0056] 图5为本发明实施例提供的一种游戏渲染装置的结构示意图;
- [0057] 图6为本发明实施例提供的一种游戏渲染终端的结构示意图。

## 具体实施方式

[0058] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述。

[0059] 游戏渲染是指计算机绘制游戏的场景画面的一系列处理过程,包括空间光照计算、光栅化和剪裁、纹理采样、深度检测、透明混合等过程。游戏渲染需要使用各类资源,包括动画、纹理贴图等等。其中,纹理贴图是指游戏的运行场景中使用的贴图文件,其能够使得游戏的场景画面更为真实;纹理贴图有各种不同类型的格式,也有各种应用场景,例如:有贴在3D (3Dimension,三维) 模型上或者用于立体空间UI (User's Interface, 用户界面) 交互上的3D纹理贴图,也有贴在2D平面上或者用于平面UI交互上的2D纹理贴图。纹理采样一般分为最近点采样和线性采样,最近点采样是根据纹理贴图的像素坐标对纹理贴图包含的原始纹理数据进行采样,找到最接近的原始纹理数据,使用最接近的原始纹理数据的像素作为纹理贴图的颜色值;线性采样是在找到最接近的原始纹理数据之后,在这个原始纹理数据附近继续采样多个数据并计算所采样的各个数据的像素平均值,将该像素平均值确定为纹理贴图的颜色值。

[0060] 一般地,游戏渲染所用到的诸如纹理贴图等资源是在游戏开发过程中由美术制作而成,制作得到的原始资源并不能直接在游戏的应用程序中使用,而是需要经过一些处理,包括二进制化、加密压缩等,这些处理过程称之为资源打包过程,所获得的文件则是游戏的资源文件,该资源文件中包含游戏渲染所需的动画、纹理贴图等资源,且资源文件中的资源可直接应用至游戏中。

[0061] 传统的游戏渲染方法,一般是预先将游戏的当前运行场景中可能需要使用的所有纹理贴图(如景物、角色和UI等)全部加载至内存中,当需要对某一纹理贴图进行渲染时,从内存中直接获取该纹理贴图,然而很多纹理贴图其实并不是一直在使用,例如:场景中的某种兵器可能并不是一开始就出现,不需要一开始就渲染,将其加载至内存中,既增加了加载开销,也增加了内存占用,进而造成游戏卡顿。现有技术中还出现另一种游戏渲染方法,能够一定程度上减小内存负担,该方法使用纹理合并工具(如texture packer工具),在资源打包过程中使用texture packer把多张小纹理贴图合并到一张大纹理贴图中,在需要对多个游戏元素(如景物、角色等)进行渲染时才将该大纹理贴图加载至内存中,从而节省一定加载开销;在渲染多个游戏元素时只要这些元素使用的是同一张大纹理贴图,并且渲染参数一致便可以合并渲染批次,从而提升渲染性能。这种使用纹理合并工具的方法虽然可以

提高渲染性能,但需要整张大纹理贴图同时加载到内存,由于游戏运行时存在着大量的纹理复用,这样的方式同样会造成大量的不必要的内存开销,并且随着游戏版本更新和内容增加而使得内存负担显著上升,进而造成游戏卡顿。

[0062] 本发明实施例在资源打包过程中将纹理贴图碎化处理为多块碎化纹理,按照游戏渲染需要加载所用的目标纹理贴图对应的碎化纹理,能够有效的降低内存加载开销,减小内存负担;另外,对目标纹理贴图对应的碎化纹理进行合并渲染,能够保证渲染画面效果,避免内存负担随游戏运行线性增加,从而保证游戏运行顺畅。

[0063] 基于上述描述,本发明实施例提供了一种游戏渲染方法,请参见图1,该游戏渲染方法可包括以下步骤S101-S104。

[0064] S101,获取游戏的资源文件,所述资源文件包括游戏渲染所需的至少一张纹理贴图,其中,一张所述纹理贴图由多块碎化纹理合并而成。

[0065] 游戏的资源文件是经资源打包过程处理后形成的文件,其包括但不限于游戏渲染所需的一张或多张纹理贴图,还可以包含游戏渲染所需的一些动画资源;此处需要特别说明的是,资源文件中的纹理贴图并非传统的整张纹理贴图,而是由多块碎化纹理拼接合并而成。通常,游戏的资源文件存储在游戏所在的游戏渲染终端(以下简称“终端”的存储空间中,例如:电脑游戏的资源文件存储于电脑为游戏所分配的存储空间中;或者,手机游戏的资源文件存储于手机为游戏所分配的存储空间中;那么,可以从游戏所在的终端的存储空间中获取游戏的资源文件。

[0066] S102,确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图。

[0067] 游戏通常会用网格来呈现与区域相关的元素,例如:某游戏场景的某个区域需要呈现地图,那么可基于网格来呈现地图这一元素。常见的网格包括方形网格、六角网格、三角网格等等。一个游戏场景画面通常会包含多个网格,当前需要在某个网格呈现某个元素(如角色、景物等),则需要对该网格进行渲染,此时该网格即为待渲染网格。游戏的场景画面中的网格与所用的纹理贴图在游戏渲染之前已确定,例如:游戏在开发过程中即确定某场景画面中的某个位置应当出现哪个元素,则该场景画面、网格与该元素对应的纹理贴图相关联,具体可以通过场景画面的序号、网格所在像素坐标范围以及纹理贴图ID相关联。因此步骤S102中可以根据游戏的当前运行场景以及待渲染网格来确定所用的目标纹理贴图。

[0068] S103,从所述资源文件中查找所述目标纹理贴图对应的碎化纹理。

[0069] 资源文件中的目标纹理贴图并非一整张纹理贴图,而是由多块碎化纹理拼接合并而成;那么,在确定当前需要渲染目标纹理贴图时,可从资源文件中查找到该目标纹理贴图对应的多块碎化纹理。

[0070] S104,对所述目标纹理贴图对应的碎化纹理进行合并渲染。

[0071] 对目标纹理贴图的多块碎化纹理进行合并渲染,而不对整张目标纹理贴图进行渲染,其目的在于降低显存和提高渲染性能。具体地,合并渲染的过程可包括:首先将目标纹理贴图的多块碎化纹理加载至内存,待加载完成后提交至游戏渲染引擎进行渲染。实际应用中,对多块碎化纹理进行合并渲染所得到的渲染画面效果与对整张目标纹理贴图进行渲染所得到的渲染画面效果保持完全一致,但内存及显存压力明显减小,渲染性能得到了有效提升。

[0072] 本发明实施例在资源打包过程中将纹理贴图碎化处理为多块碎化纹理,按照游戏

渲染需要加载所用的目标纹理贴图对应的碎化纹理，能够有效的降低内存加载开销，减小内存负担；另外，对目标纹理贴图对应的碎化纹理进行合并渲染，能够保证渲染画面效果，避免内存负担随游戏运行线性增加，从而保证游戏运行顺畅。

[0073] 本发明实施例还提供了另一种游戏渲染方法，请参见图2，该游戏渲染方法可包括以下步骤S201-S210。

[0074] S201，获取游戏渲染所需的至少一张纹理贴图，所述纹理贴图包含原始纹理数据。

[0075] S202，分别将各纹理贴图碎化处理为多块碎化纹理，一块碎化纹理包括一个碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据。

[0076] S203，将各纹理贴图对应的碎化纹理封装至资源文件中。

[0077] 步骤S201-S203属于资源打包过程，游戏开发时由美术制作游戏渲染所需的所有纹理贴图，这些纹理贴图以原始纹理数据的形式存在，在资源打包的过程中，对纹理贴图的原理纹理数据进行碎化，一张纹理贴图可以被碎化为多块碎化纹理，这些碎化纹理被封装至资源文件中，供游戏渲染时使用。

[0078] 其中，步骤S202中的碎化处理过程具体可包括以下步骤s11-s13：

[0079] s11，依次选取各张纹理贴图，将所选纹理贴图按照第一预设规格进行切割得到多个矩形框；

[0080] s12，对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多个碎化纹理数据，并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据；

[0081] s13，将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框，一个完成填充的矩形框形成一块碎化纹理。

[0082] 具体实现中，步骤s13可以包括以下过程：首先，在每个矩形框上下两侧各预留预设宽度的边缘，并在每个矩形框左右两侧各增加预设宽度的边缘，使每个矩形框具备中央区和包围中央区的边缘区；其次，将多个碎化纹理数据分别填充至对应矩形框的中央区，并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区，一个完成填充的矩形框形成一块碎化纹理。

[0083] 每张纹理贴图重复上述步骤s11-s13进行碎化处理，可将所有纹理贴图均碎化为多块碎化处理。其中，第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和；例如：第一预设规格的值为32像素，第二预设规格的值为30像素，预设宽度为1像素；或者，第一预设规格的值为16像素，第二预设规格的值14像素，预设宽度为1像素。

[0084] 下面结合附图3对步骤s11-s13的碎化处理过程进行介绍。假设第一预设规格为32像素，将纹理贴图按32像素的行高进行分割可得到图3a所示的多个矩形框，每个矩形框的行高均为32像素。一张纹理贴图的多块碎化纹理在显存中的位置并非连续有序的，其可能是分散的，当多块碎化纹理重新拼接为一张纹理贴图并进行渲染的过程中，如果进行线性采样便会出现不规则的颜色接缝；为了避免线性采样出现不规则的颜色接缝，本实施例碎化处理过程需要进行特殊处理，该特殊处理过程可结合附图3b-3c，具体为：当得到多个行高为32像素的矩形框之后，在每个矩形框上下两侧各预留1像素的边缘，并在每个矩形框左右两侧各增加1像素的边缘，使每个矩形框具备高度为30像素的中央区和包围中央区的宽度为1像素的边缘区；然后对纹理贴图所包含的原始纹理数据按照30像素行高进行碎化处理得到多个碎化纹理数据，并从纹理贴图中获得各个碎化纹理数据的周边1像素宽度的边

缘原始纹理数据；将行高为30像素的碎化纹理数据填充至矩形框的中央区，并在四周1像素宽度的边缘区内填充边缘原始纹理数据，这样一个完成填充的矩形框形成一块碎化纹理。

[0085] 其中，步骤S203中的封装过程具体可包括以下步骤s21-s23：

[0086] s21，获取各纹理贴图的属性，所述属性包括ID、释放类型及偏移信息，所述释放类型包括立即释放、自动释放或常驻内存；

[0087] 一张纹理贴图的ID用于唯一标识一张纹理贴图。偏移信息用于确定纹理贴图在资源文件中的存储位置。释放类型可以在游戏渲染之前在游戏的渲染引擎中设置；其中：

[0088] 立即释放是指：无论是增加或者更换纹理贴图，对于纹理贴图的这类访问均不会触发该纹理贴图的加载，只有在真正需要用到该纹理贴图时（比如在游戏场景中绘制某元素需要使用该纹理贴图）才会对该纹理贴图进行加载；并且，该纹理贴图加载完成后，一旦不再使用该纹理贴图时（比如元素销毁或隐藏时）立即释放该纹理贴图所占用的内存空间；并且，对纹理贴图的释放采用延时处理，如延时2s释放或延时5s释放，防止该纹理贴图被释放后立即需要加载的情况，通常游戏外界面所使用的纹理贴图为立即释放类型。

[0089] 自动释放是指：为已加载的纹理贴图设置引用计数，类似于智能指针，该纹理贴图被访问一次则增加引用计数，而该纹理贴图对应的元素销毁或该纹理贴图被更换则减少引用计数，一旦引用计数归零则自动释放该纹理贴图所占用的内存空间。

[0090] 常驻内存是指：纹理贴图被加载至内存后，只能使用游戏渲染引擎内部接口强制释放该纹理贴图所占用的内存空间。

[0091] s22，根据各纹理贴图的属性生成简要信息表；

[0092] s23，将所述简要信息表存放至资源文件的文件头，将各纹理贴图对应的碎化纹理存放至资源文件的文件主体。

[0093] 步骤s21-s23中，资源文件由两部分组成，文件头中存放了简要信息表，而文件主体存放了纹理贴图对应的多块碎化纹理；通过简要信息表能够实现对纹理贴图及其碎化纹理的按需查找及按需加载。

[0094] S204，获取游戏的资源文件，所述资源文件包括游戏渲染所需的至少一张纹理贴图，其中，一张所述纹理贴图由多块碎化纹理合并而成。

[0095] S205，确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图。

[0096] S206，从所述资源文件中查找所述目标纹理贴图对应的碎化纹理。

[0097] S207，对所述目标纹理贴图对应的碎化纹理进行合并渲染。

[0098] 本实施例的步骤S204-S207可参见图1所示实施例的步骤S101-S104，其中步骤S207的合并渲染的过程具体包括以下步骤s31-s34：

[0099] s31，按照第一预设规格将内存中的预置动态加载区进行切割，形成多个动态加载行；

[0100] s32，采用装箱算法对所述动态加载行的空闲区域进行分配；

[0101] s33，依次将所述目标纹理贴图对应的碎化纹理加载至所述动态加载行的空闲区域；

[0102] s34，从所述动态加载行中读取所加载的碎化纹理并提交至游戏的渲染引擎进行渲染。

[0103] 步骤s31-s34中，内存中的预置动态加载区是指在内存中预分配的、用于动态加载

碎化纹理的区域,根据实际情况,预置动态加载区可以为一张或若干张动态纹理贴图的大小,例如:预置动态加载区可以表现为一张或若干张大小为2048\*1024的动态纹理贴图。可按照第一预设规格(16像素或32像素)将内存中的预置动态加载区切割成若干行,每一行即为一个动态加载行;针对每一个动态加载行可以采用装箱算法来对其中的空闲区域进行分配,将空闲区域分配给需要加载的目标纹理贴图的多块碎化纹理,待这些碎化纹理被加载完成后,将这些碎化纹理提交至游戏的渲染引擎进行渲染。需要说明的是,此处的合并渲染过程采用渲染优化算法,该渲染优化算法以现有整张纹理贴图的渲染方式和CPU剪裁为基础,利用现有渲染方式计算整张纹理贴图的坐标,然后每块碎化后纹理在整张纹理贴图中的相对位置比例计算每块碎化纹理的坐标。具体实现中,该渲染优化算法可采用以下代码片段表示:

[0104]

```
blend=(smash->GetFormat()==NPPixelFormat::PF_R8G8B8A8)||srcabm !=0||bcolor.a  
!=1.0f);  
  
nfp32 sx1=(nfp32)(smash->GetImageX());  
nfp32 sy1=(nfp32)(smash->GetImageY());  
nfp32      sx2=NPMathWrap::Min(sx1+(nfp32)(smash->GetImageW()-2)      ,  
(nfp32)(smash->GetSrcX()+smash->GetImage()->GetSrcW()));  
nfp32      sy2=NPMathWrap::Min(sy1+(nfp32)(NP2DSSmash::HTILE-2)      ,  
(nfp32)(smash->GetSrcX()+smash->GetImage()->GetSrcW()));  
  
if (uv2.x<sx1||  
    uv1.x>sx2||  
    uv2.y<sy1||  
    uv1.y>sy2)  
    return;  
texID=smash->GetSmashTexture()->GetTextureID();
```

[0105]

```
NPTexture*texture=NP2DSTextureCache::GetIns()->GetTexture(texID);  
nfp32 invw=1.0f/texture->GetWidth();  
nfp32 invh=1.0f/texture->GetHeight();  
  
nfp32 fx1=(uv1.x>sx1)?uv1.x:sx1;  
nfp32 fy1=(uv1.y>sy1)?uv1.y:sy1;  
nfp32 fx2=(uv2.x<sx2)?uv2.x:sx2;  
nfp32 fy2=(uv2.y<sy2)?uv2.y:sy2;  
  
nfp32 rx1=(fx1-uv1.x)/(uv2.x-uv1.x);  
nfp32 ry1=(fy1-uv1.y)/(uv2.y-uv1.y);  
nfp32 rx2=(fx2-uv1.x)/(uv2.x-uv1.x);  
nfp32 ry2=(fy2-uv1.y)/(uv2.y-uv1.y);  
  
nfp32 dx1;  
nfp32 dy1;  
nfp32 dx2;  
nfp32 dy1;  
if (NPConfig::GetGlobalCoef()==0)  
{  
    dx1=(nfp32)(smash->GetSmashX()+1)+(fx1-sx1)/2.0f;  
    dy1=(nfp32)(smash->GetSmashR()*NP2DSSmash::HCOEF+1)+(fy1-sy1)/2.0f;  
    dx2=dx1+(fx2-fx1)/2.0f;  
    dy2=dy1+(fy2-fy1)/2.0f;  
}  
  
npu32 abgr=bcolor.GetABGR();  
NP2DSRenderPool::Vertex vertexs[4];  
NPVector2 pt1=pt+d1*(l1*rx1)+d2*(l2*ry1);  
NPVector2 pt2=pt+d1*(l1*rx2)+d2*(l2*ry2);
```

- ```
NPVector2 pt3=pt+d1*(l1*rx1)+d2*(l2*ry2);  
[0106] NPVector2 pt4=pt+d1*(l1*rx2)+d2*(l2*ry1);  
[0107] 请参见图4所示游戏中的实际渲染效果以及碎化纹理在显存中的存储状态,可以看出显存中的碎化纹理是按每行顺序进行排列的。实际应用中,对多块碎化纹理进行合并渲染所得到的渲染画面效果与对整张目标纹理贴图进行渲染所得到的渲染画面效果保持完全一致,但内存及显存压力明显减小,渲染性能得到了有效提升。  
[0108] S208,从资源文件的文件头中获取所述目标纹理贴图的释放类型。  
[0109] S209,按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行。  
[0110] S210,将回收的动态加载行进行合并。  
[0111] 步骤S208-S210中,在对目标纹理贴图的多块碎化纹理进行合并渲染之后,需要根据目标纹理贴图的释放类型来释放各块碎化纹理所占用的内存空间;具体地,由于碎化纹理被加载至内存中的动态加载行中,在释放碎化纹理所占用的内存空间后,动态加载行回收其分配出去的空闲区域,回收后的动态加载行再进行合并,可以理解的是,各个动态加载行最初是由预置动态加载区切割得到的,待所有动态加载行被回收后,所有动态加载行又可合并形成预置动态加载区。  
[0112] 本发明实施例在资源打包过程中将纹理贴图碎化处理为多块碎化纹理,按照游戏渲染需要加载所用的目标纹理贴图对应的碎化纹理,能够有效的降低内存加载开销,减小内存负担;另外,对目标纹理贴图对应的碎化纹理进行合并渲染,能够保证渲染画面效果,避免内存负担随游戏运行线性增加,从而保证游戏运行顺畅。  
[0113] 基于上述的游戏渲染方法的实施例,本发明实施例还公开了一种游戏渲染装置,该游戏渲染装置可以是一个计算机程序(包括程序代码),且该计算机程序可以运行于终端(如PC(Personal Computer,个人计算机)、手机等)中以用来执行图1-2任一实施例所示的游戏渲染方法。请参见图5,该游戏渲染装置运行如下单元:  
[0114] 文件获取单元101,用于获取游戏的资源文件,所述资源文件包括游戏渲染所需的至少一张纹理贴图,各纹理贴图分别由多块碎化纹理合并而成。  
[0115] 目标确定单元102,用于确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图。  
[0116] 查找单元103,用于从所述资源文件中查找所述目标纹理贴图对应的多块碎化纹理。  
[0117] 合并渲染单元104,用于对所述目标纹理贴图的多块碎化纹理进行合并渲染。  
[0118] 具体实现中,所述游戏渲染装置还运行如下单元:  
[0119] 资源打包处理单元105,用于获取游戏渲染所需的至少一张纹理贴图,所述纹理贴图包含原始纹理数据,分别将各纹理贴图碎化处理为多块碎化纹理,一块碎化纹理包括一个碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据,以及将各纹理贴图对应的多块碎化纹理封装至资源文件中;以及,  
[0120] 回收处理单元106,用于从资源文件的文件头中获取所述目标纹理贴图的释放类型,按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行,
```

以及将回收的动态加载行进行合并。

[0121] 具体实现中,该游戏渲染装置运行所述资源打包处理单元105的过程具体为:依次选取各张纹理贴图,将所选纹理贴图按照第一预设规格进行切割得到多个矩形框;对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多个碎化纹理数据,并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据;将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理;其中,第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和;

[0122] 其中,将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框,一个完成填充的矩形框形成一块碎化纹理包括:在每个矩形框上下两侧各预留预设宽度的边缘,并在每个矩形框左右两侧各增加预设宽度的边缘,使每个矩形框具备中央区和包围中央区的边缘区;将多个碎化纹理数据分别填充至对应矩形框的中央区,并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区,一个完成填充的矩形框形成一块碎化纹理;以及,

[0123] 具体为:获取各纹理贴图的属性,所述属性包括ID、释放类型及偏移信息,所述释放类型包括立即释放、自动释放或常驻内,根据各纹理贴图的属性生成简要信息表,以及将所述简要信息表存放至资源文件的文件头,将各纹理贴图对应的多块碎化纹理存放至资源文件的文件主体。

[0124] 具体实现中,该游戏渲染装置运行所述合并渲染单元104的过程具体为:按照第一预设规格将内存中的预置动态加载区进行切割,形成多个动态加载行;采用装箱算法对所述动态加载行的空闲区域进行分配;依次将所述目标纹理贴图对应的各块碎化纹理加载至所述动态加载行的空闲区域;以及,从所述动态加载行中读取所加载的碎化纹理并提交至游戏的渲染引擎进行渲染。

[0125] 根据本发明的一个实施例,图1所示的游戏渲染方法涉及的步骤S101-S104可以是由图5所示的游戏渲染装置中的各个单元来执行的。例如,图1中所示的步骤S101-S104可以分别由图5中所示的文件获取单元101、目标确定单元102、查找单元103和合并渲染单元104来执行。

[0126] 根据本发明的另一个实施例,图2所示的游戏渲染方法涉及的步骤S201-S210可以是由图5所示的游戏渲染装置中的各个单元来执行的。例如,图2中所示的步骤S201-S203、S204、S205、S206、S207、S208-S210可以分别由图5中所示的资源打包处理单元105、文件获取单元101、目标确定单元102、查找单元103、合并渲染单元104和回收处理单元106来执行。

[0127] 根据本发明的另一个实施例,图5所示的游戏渲染装置中的各个单元可以分别或全部合并为一个或若干个另外的单元来构成,或者其中的某个(些)单元还可以再拆分为功能上更小的多个单元来构成,这可以实现同样的操作,而不影响本发明的实施例的技术效果的实现。上述单元是基于逻辑功能划分的,在实际应用中,一个单元的功能也可以由多个单元来实现,或者多个单元的功能由一个单元实现。在本发明的其它实施例中,游戏渲染装置也可以包括其它单元,在实际应用中,这些功能也可以由其它单元协助实现,并且可以由多个单元协作实现。

[0128] 根据本发明的另一个实施例,可以通过在包括中央处理单元(CPU)、随机存取存储

设备(RAM)、只读存储设备(ROM)等处理元件和存储元件的例如计算机的通用计算设备上运行能够执行如图1-图2中所示的游戏渲染方法涉及的各步骤的计算机程序(包括程序代码),来构造如图5中所示的游戏渲染装置设备,以及来实现本发明实施例的游戏渲染方法。所述计算机程序可以记载于例如计算机可读记录介质上,并通过计算机可读记录介质装载于上述计算设备中,并在其中运行。

[0129] 本发明实施例在资源打包过程中将纹理贴图碎化处理为多块碎化纹理,按照游戏渲染需要加载所用的目标纹理贴图对应的碎化纹理,能够有效的降低内存加载开销,减小内存负担;另外,对目标纹理贴图对应的碎化纹理进行合并渲染,能够保证渲染画面效果,避免内存负担随游戏运行线性增加,从而保证游戏运行顺畅。

[0130] 基于上述实施例所示的游戏渲染方法及游戏渲染装置,本发明实施例还提供了一种终端,该终端可用于执行上述图1-图2所示方法流程的相应步骤。具体实现中,本发明实施例中描述的终端包括但不限于诸如具有触摸敏感表面(例如,触摸屏显示器和/或触摸板)的移动电话、膝上型计算机或平板计算机之类的其它便携式设备。还应当理解的是,在某些实施例中,所述设备并非便携式通信设备,而是具有触摸敏感表面(例如,触摸屏显示器和/或触摸板)的台式计算机。请参见图6,该终端的内部结构可包括处理器、用户接口、网络接口及存储设备。其中,终端内的处理器、用户接口、网络接口及存储设备可通过总线或其他方式连接,在本发明实施例所示图6中以通过总线连接为例。

[0131] 其中,用户接口是实现用户与终端进行交互和信息交换的媒介,其具体体现可以包括用于输出的显示屏(Display)以及用于输入的键盘(Keyboard)等等,需要说明的是,此处的键盘既可以为实体键盘,也可以为触屏虚拟键盘,还可以为实体与触屏虚拟相结合的键盘。然而,应当理解的是,用户接口还可以包括诸如鼠标和/或控制杆的一个或多个其它物理用户接口设备。处理器(或称CPU(Central Processing Unit,中央处理器))是终端的计算核心以及控制核心,其适于实现一条或一条以上指令,具体适于加载并执行一条或一条以上指令从而实现相应方法流程或相应功能;例如:CPU可以用于解析用户向终端所发送的开关机指令,并控制终端进行开关机操作;再如:CPU可以在终端内部结构之间传输各类交互数据,等等。存储设备(Memory)是终端中的记忆设备,用于存放程序和数据。可以理解的是,此处的存储设备既可以包括终端的内置存储设备,当然也可以包括终端所支持的扩展存储设备。存储设备提供存储空间,该存储空间存储了终端的操作系统。并且,在该存储空间中还存放了适于被处理器加载并执行的一条或一条以上的指令,这些指令可以是一个或一个以上的计算机程序(包括程序代码)。需要说明的是,此处的存储设备可以是高速RAM存储器,也可以是非不稳定的存储器(non-volatile memory),例如至少一个磁盘存储器;可选的还可以是至少一个位于远离前述处理器的存储设备。

[0132] 还需要特别说明的是,终端支持各种应用程序,例如以下中的一个或多个:绘图应用程序、演示应用程序、文字处理应用程序、网站创建应用程序、盘刻录应用程序、电子表格应用程序、游戏应用程序、电话应用程序、视频会议应用程序、电子邮件应用程序、即时消息收发应用程序、锻炼支持应用程序、照片管理应用程序、数码相机应用程序、数字摄影机应用程序、web浏览应用程序、数字音乐播放器应用程序和/或数字视频播放器应用程序。可以在终端上执行的各种应用程序可以使用诸如触摸敏感表面的至少一个公共物理用户接口设备。可以在应用程序之间和/或相应应用程序内调整和/或改变触摸敏感表面的一个或多

个功能以及终端上显示的相应信息。这样，终端的公共物理架构(例如，触摸敏感表面)可以支持具有对用户而言直观且透明的用户界面的各种应用程序。

[0133] 在本发明实施例中，处理器加载并执行存储设备中存放的一条或一条以上指令，以实现上述图1-图2所示方法流程的相应步骤；具体实现中，存储设备中的一条或一条以上指令由处理器加载并执行如下步骤：

[0134] 获取游戏的资源文件，所述资源文件包括游戏渲染所需的至少一张纹理贴图，其中，一张所述纹理贴图由多块碎化纹理合并而成；

[0135] 确定游戏的当前运行场景中待渲染网格所用的目标纹理贴图；

[0136] 从所述资源文件中查找所述目标纹理贴图对应的碎化纹理；

[0137] 对所述目标纹理贴图对应的碎化纹理进行合并渲染。

[0138] 具体实现中，所述处理器加载存储设备中的一条或一条以上指令在执行获取游戏的资源文件这一步骤之前，还执行如下步骤：

[0139] 获取游戏渲染所需的至少一张纹理贴图，所述纹理贴图包含原始纹理数据；

[0140] 分别将各纹理贴图碎化处理为多块碎化纹理，一块碎化纹理包括一个碎化纹理数据以及包围所述碎化纹理数据的预设宽度的边缘原始纹理数据；

[0141] 将各纹理贴图对应的碎化纹理封装至资源文件中。

[0142] 具体实现中，所述处理器加载存储设备中的一条或一条以上指令在执行分别将各纹理贴图碎化处理为多块碎化纹理这一步骤的过程中，具体执行如下步骤：

[0143] 依次选取各张纹理贴图，将所选纹理贴图按照第一预设规格进行切割得到多个矩形框；

[0144] 对所选纹理贴图所包含的原始纹理数据按照第二预设规格进行碎化处理得到多个碎化纹理数据，并在各个碎化纹理数据的四周获取预设宽度的边缘原始纹理数据；

[0145] 将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框，一个完成填充的矩形框形成一块碎化纹理；

[0146] 其中，第一预设规格的值大于或等于第二预设规格的值与两倍预设宽度的值的总和。

[0147] 具体实现中，所述处理器加载存储设备中的一条或一条以上指令在执行将多个碎化纹理数据及预设宽度的边缘原始纹理数据填充至对应矩形框，一个完成填充的矩形框形成一块碎化纹理这一步骤的过程中，具体执行如下步骤：

[0148] 在每个矩形框上下两侧各预留预设宽度的边缘，并在每个矩形框左右两侧各增加预设宽度的边缘，使每个矩形框具备中央区和包围中央区的边缘区；

[0149] 将多个碎化纹理数据分别填充至对应矩形框的中央区，并将包围各个碎化纹理数据的预设宽度的边缘原始纹理数据分别填充至对应矩形框的边缘区，一个完成填充的矩形框形成一块碎化纹理。

[0150] 具体实现中，所述处理器加载存储设备中的一条或一条以上指令在执行所述将各纹理贴图对应的碎化纹理封装至资源文件中这一步骤的过程中，具体执行如下步骤：

[0151] 获取各纹理贴图的属性，所述属性包括ID、释放类型及偏移信息，所述释放类型包括立即释放、自动释放或常驻内存；

[0152] 根据各纹理贴图的属性生成简要信息表；

[0153] 将所述简要信息表存放至资源文件的文件头,将各纹理贴图对应的碎化纹理存放至资源文件的文件主体。

[0154] 具体实现中,所述处理器加载存储设备中的一条或一条以上指令在执行所述对所述目标纹理贴图对应的碎化纹理进行合并渲染这一步骤的过程中,具体执行如下步骤:

[0155] 按照第一预设规格将内存中的预置动态加载区进行切割,形成多个动态加载行;

[0156] 采用装箱算法对所述动态加载行的空闲区域进行分配;

[0157] 依次将所述目标纹理贴图对应的碎化纹理加载至所述动态加载行的空闲区域;

[0158] 从所述动态加载行中读取所加载的碎化纹理并提交至游戏的渲染引擎进行渲染。

[0159] 具体实现中,所述处理器加载存储设备中的一条或一条以上指令在执行所述对所述目标纹理贴图的多块碎化纹理进行合并渲染这一步骤之后,还执行如下步骤:

[0160] 从资源文件的文件头中获取所述目标纹理贴图的释放类型;

[0161] 按照所获取的释放类型释放所述目标纹理贴图的各块碎化纹理所占用的动态加载行;

[0162] 将回收的动态加载行进行合并。

[0163] 本发明实施例在资源打包过程中将纹理贴图碎化处理为多块碎化纹理,按照游戏渲染需要加载所用的目标纹理贴图对应的碎化纹理,能够有效的降低内存加载开销,减小内存负担;另外,对目标纹理贴图对应的碎化纹理进行合并渲染,能够保证渲染画面效果,避免内存负担随游戏运行线性增加,从而保证游戏运行顺畅。

[0164] 在本说明书的描述中,参考术语“一个实施例”、“一些实施例”、“示例”、“具体示例”、或“一些示例”等的描述意指结合该实施例或示例描述的具体特征、结构、材料或者特点包含于本发明的至少一个实施例或示例中。在本说明书中,对上述术语的示意性表述不必针对的是相同的实施例或示例。而且,描述的具体特征、结构、材料或者特点可以在任一个或多个实施例或示例中以合适的方式结合。此外,在不相互矛盾的情况下,本领域的技术人员可以将本说明书中描述的不同实施例或示例以及不同实施例或示例的特征进行结合和组合。

[0165] 此外,术语“第一”、“第二”仅用于描述目的,而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此,限定有“第一”、“第二”的特征可以明示或者隐含地包括至少一个该特征。在本发明的描述中,“多个”的含义是至少两个,例如两个,三个等,除非另有明确具体的限定。

[0166] 流程图中或在此以其他方式描述的任何过程或方法描述可以被理解为,表示包括一个或更多个用于实现特定逻辑功能或过程的步骤的可执行指令的代码的模块、片段或部分,并且本发明的实施方式的范围包括另外的实现,其中可以不按所示出或讨论的顺序,包括根据所涉及的功能按基本同时的方式或按相反的顺序,来执行功能,这应被本发明的实施例所属技术领域的技术人员所理解。

[0167] 应当理解,本发明的各部分可以用硬件、软件、固件或它们的组合来实现。在上述实施方式中,多个步骤或方法可以用存储在存储设备中且由合适的指令执行系统执行的软件或固件来实现。例如,如果用硬件来实现,和在另一实施方式中一样,可用本领域公知的下列技术中的任一项或他们的组合来实现:具有用于对数据信号实现逻辑功能的逻辑门电

路的离散逻辑电路,具有合适的组合逻辑门电路的专用集成电路,可编程门阵列(PGA),现场可编程门阵列(FPGA)等。此外,在本发明各个实施例中的各功能单元可以集成在一个处理模块中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个模块中。上述集成的模块既可以采用硬件的形式实现,也可以采用软件功能模块的形式实现。所述集成的模块如果以软件功能模块的形式实现并作为独立的产品销售或使用时,也可以存储在一个计算机可读取存储介质中。

[0168] 以上所揭露的仅为本发明较佳实施例而已,当然不能以此来限定本发明之权利范围,因此依本发明权利要求所作的等同变化,仍属本发明所涵盖的范围。

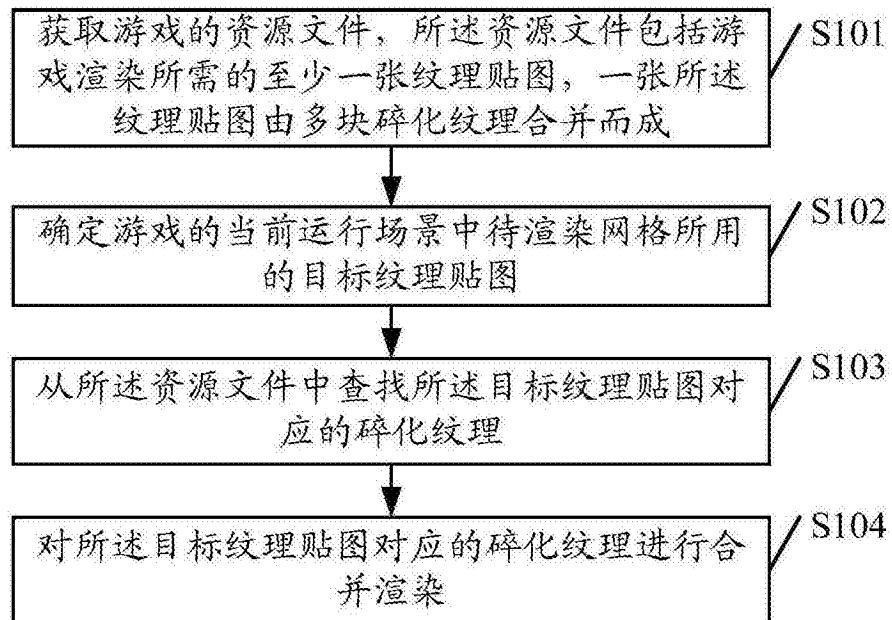


图1



图2



图3a

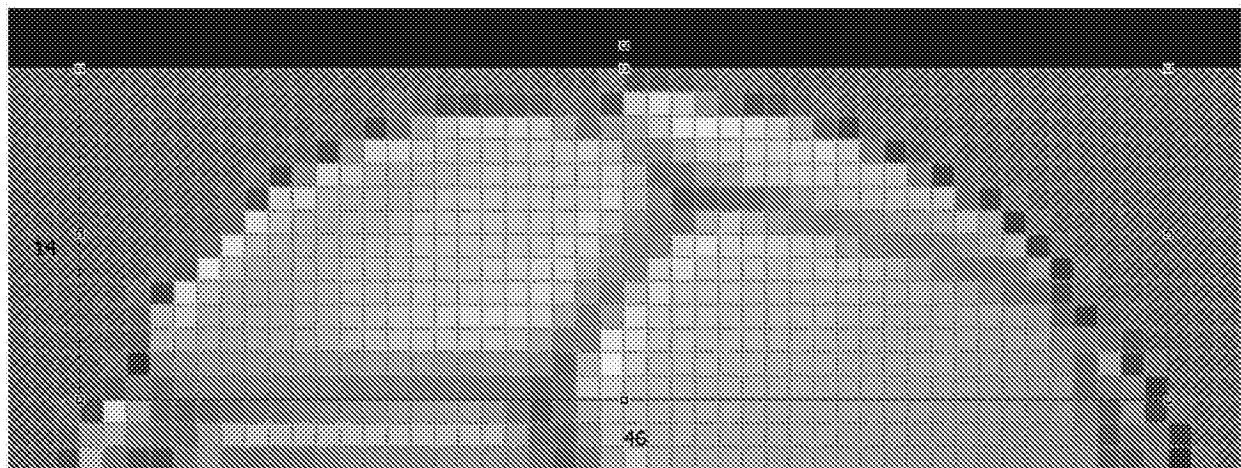


图3b

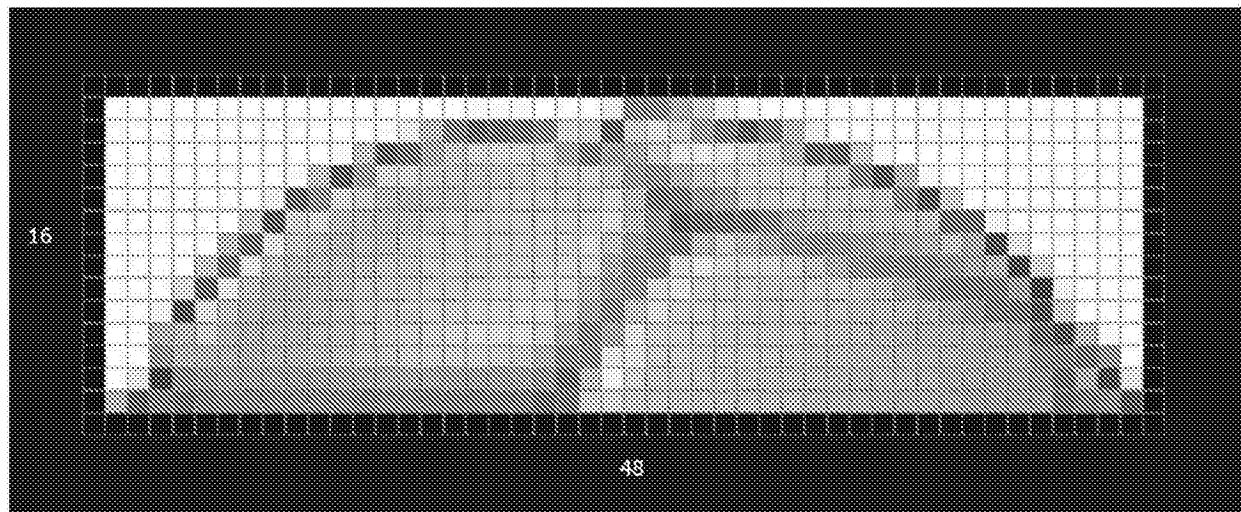


图3c



图4

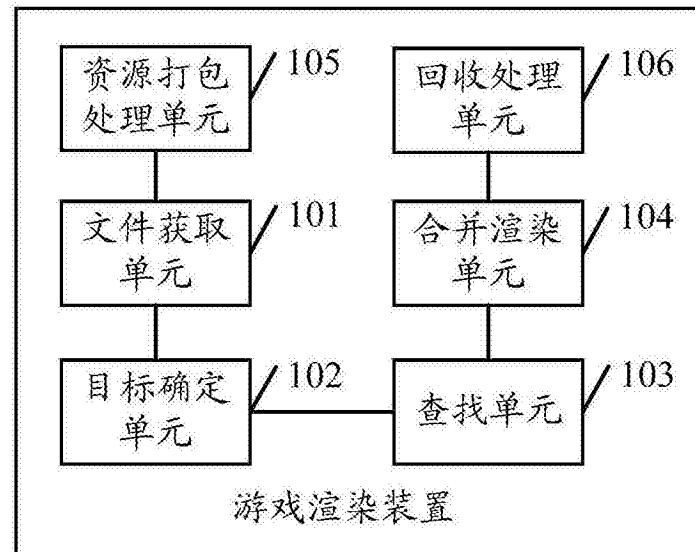


图5

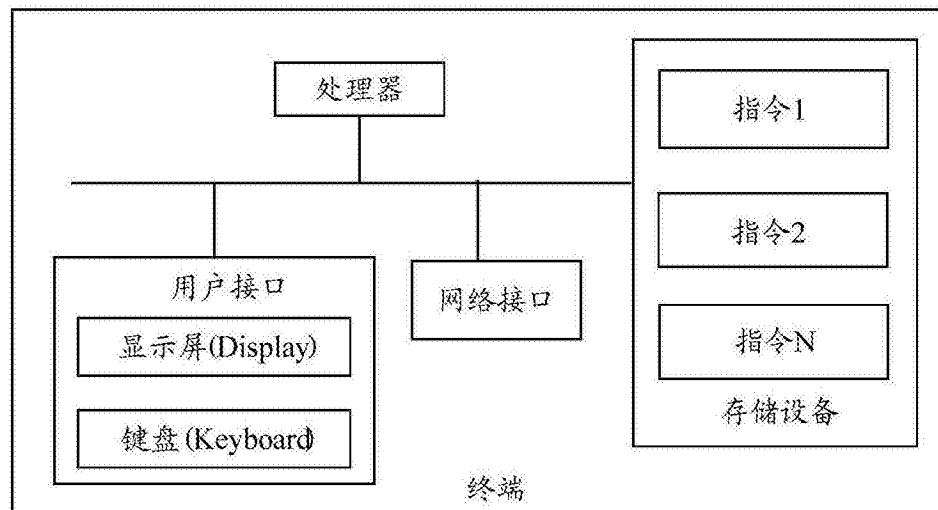


图6