云渲染测试方案

1. 数据准备（1小时）

以下参数皆用于ModelGenerator，详见附录相关说明。

* 1. 通用参数：--cylinder 150000 --srand 123 --spanY 330000 --spaceScale 1.5
  2. 计算显存模型：--equipCol 1 --equipRow 1
  3. 车间级人员数据：--equipCol 2 --equipRow 3
  4. 厂区级人员数据：--equipCol 4 --equipRow 5
  5. 全厂级人员数据：--equipCol 10 --equipRow 20

1. 录制测试脚本（？天）
   1. 什么是复杂场景：任何一个装置都是复杂场景，需要让视角在任意一个装置附近，朝向装置中心附近位置即可。
   2. 测试脚本：
      1. 单项测试

* **旋转**：先定位到某个复杂场景，以当前装置中心为视角旋转中心，以Z轴为旋转轴，横向旋转场景一周。可以分多次。
* **平移**：先定位到某个复杂场景，先向左移动视角，直到当前装置完全移出视野，再向右移动视角，直到当前装置完全进入视野，停3秒左右到模型全部或近乎全部加载完毕，再从头重复操作。
* **缩放**：先定位到某个复杂场景，使用鼠标滚轮先缩小模型，直到当前装置消隐大半，再使用滚轮放大模型，放大到最开始的视角，停3秒左右到模型全部或近乎全部加载完毕，再从头重复操作。
* **高亮**：先定位到某个复杂场景，点选任何一个模型，使其高亮。应该高亮得多才更有性能压力。
  + 1. 综合测试

通过交替使用旋转、平移、缩放和高亮，来使场景不断在不同复杂场景之前来回切换，并注意在场景有较大变动时，需要等待未加载的模型全部或近乎全部加载完毕，再进行下一步操作。

~~录制脚本时需要尽量让视角在不同的复杂场景下来回切换，不断加载和渲染不同的模型，以对系统施加最大压力，得到最准确的性能数据。~~

模型的旋转、平移、缩放、高亮操作需要各自单独操作3分钟，**分开记录**各个操作的性能数据。

在所有测试时，建议每进行一次或几次操作后有短暂的停顿（1-3秒），用以保证模型加载。

部分功能的使用（例如：目录树定位、查找等）。（某种操作的性能问题）（录制脚本原则）（脚本持续时间？）

性能压力从大到小：缩放、旋转、高亮、平移

1. 硬件准备

渲染服务器一台，两块显卡备用；10台客户端（或10台虚拟机）。

1. 软件准备

部署1台服务器，10台客户端。详见相应部署手册。

1. 数据记录
   1. 需要记录渲染服务器如下数据：

|  |  |  |
| --- | --- | --- |
| 客户端数量 |  | |
| CPU核数 |  | |
| CPU频率 |  | |
| 参数 | 平均 | 最大 |
| CPU使用率（%） |  |  |
| GPU使用率（%） |  |  |
| 内存占用（MB） |  |  |
| 显存占用（MB） |  |  |
| 带宽占用（MBPS） |  |  |
| 网络延迟（MS） |  |  |
| 磁盘平均读写速率（MBPS） |  |  |

* 1. 网络延迟测试方法

启动时不加载任何模型，调用ModelLoader.loadModelFromString添加一个简单模型，然后可以使用w键切换线框与实体模式，可以通过按键精灵来监测模型变化。

1. 测试流程
   1. V1加载模式显存占用量测定方法：

Settings.ini对应项修改如下：

[kernel]

pdsoft\_load\_version=v1

v1\_pd\_scale=100

small\_visible\_dis=8e6

middle\_visible\_dis=1e8

big\_visible\_dis=1e9

在启动平台之前，先记录显存占用量。启动平台，加载“计算显存模型”，按空格切换到俯视模式，查看显存占用量，减去之前记录的占用量，得到单个装置的显存占用量。(和实际使用场景有什么区别)

* 1. 通用测试流程

部署ClientNum个客户端，使用GCNum个显卡~~，每个用户分配GM（MB）的显存~~。将与对应等级的数据部署在服务器端并**开启性能数据记录**。在所有客户端开始播放录制好的操作脚本。

在测试过程中如果发现性能

* 1. 代入参数测试说明：首先将第7节说明的参数代入到6.2小节流程中进行实际操作。再依据当前性能参数，按照二分法调整客户端数量以得到当前硬件可以支撑的客户端数量。

1. 代入测试参数(场景、用例)(从简单开始)
   1. 车间级

|  |  |  |
| --- | --- | --- |
| 参数名 | ClientNum | GCNum |
| 值 | 2 | 1 |

* 1. 厂区级.

|  |  |  |
| --- | --- | --- |
| 参数名 | ClientNum | GCNum |
| 值 | 2 | 2 |

1. 全厂级用户测试

**附录**

1. ModelGenerator参数说明：

Allowed options:

-h [ --help ] produce help message

--db arg sqlite .db file for output

--spanX arg (=100000) spanX

--spanY arg (=200000) spanY

--spanZ arg (=10000) spanZ

--cylinderNum arg (=0) cylinder num

--sphereNum arg (=0) sphere num

--ovalNum arg (=0) oval num

--torusNum arg (=0) torus num

--srand arg (=0) random algorithm seed

--equipRow arg (=1) equipment matrix row number

--equipCol arg (=1) equipment matrix col number

--spaceScale arg (=1.5) space scale of span(X&Y) between equipments

示例：ModelGenerator --db "C:\Users\zzh\AppData\Roaming\Fulongtech Kernel\ModelCache\data\kernel.db" --equipCol 3 --equipRow 5 --cylinder 150000 --srand 123 --spanY 330000 –spaceScale 1.5

解释：在指定db生成模型，有15套装置（3×5），每套装置有15万圆柱体，每套装置之间的间隔为1.5倍。

使用上述方法生成db之后，再使用Unlink处理db后才可以使用v2和v3加载。

注：由于只生成了PDSOFT格式的模型，经Unlink处理后，v1/v2/v3都可以正常加载。

地形生成

1. 测试时的操作方式与加载模式也有关系，也与验证的瓶颈有关系。
   1. 在V1加载模式下，如果要验证对CPU、RAM、GPU的压力，则不需要什么操作，找一个较复杂场景，高亮一个模型就可以了；验证最大显存使用量则需要尽量让所有模型至少显示一次，也就需要在场景中到处游览才能得到最大内存需求量。
   2. 在V3加载模式下，主要性能问题出现在动态加载功能上。操作方式为：经常变换视角，并在变换视角后停顿一下以便模型进行加载，加载完成之后或几乎完成时就再次变换视角。
2. 各部件压力如何测定

在云桌面模式下，RAM和GM可以直接查看服务器端的各项参数。

CPU和GPU的压力除了要看服务器端的负载，也要在客户端上查看CPU和GPU各自负责的渲染阶段的处理时间，可以在客户端按再次s键，其中Event+Update+Cull+Draw为CPU处理时间，GPU为GPU处理时间。

CPU负载情况判定表：

|  |  |  |
| --- | --- | --- |
| CPU利用率  CPU处理时间 | 高 | 低 |
| 高 | 瓶颈（之一） | 算法有问题 |
| 低 | 还不错 | 完全不是问题 |

GPU同理。

1. 当各部件压力不均衡时的调整方法（应该不太容易调整到一个较平衡的负载状态）
   1. CPU：使用V1可使CPU利用率大幅下降，但内存和显存会大幅增加。
   2. RAM：可以使用V3（无妨：服务器增加内存较方便和廉价）。
   3. GPU：此时可以计算每个显卡可以支撑几个客户端（当然GM也是一个考查点）。
   4. GM：如果是贴图太大，可以使用TextureOptimizer处理；其它情况只能是使用了V1或缓存所有模型的V3模式。
2. 测试方法建议

服务器只有一个，各项参数的监测和记录只需要一个人。

客户端有多个，操作可以按键精灵来进行。如果按键精灵脚本写的好的话，数据的记录和观察只需要一到两个人，各自实际操作和监测一个客户端，进行直观的操作体验感受