# 小立方体堆叠成大立方体实现及应用说明

## 摘要

为了让堆叠效果更加逼真自然，达到较好的动态效果，这里借助物理引擎进行运动仿真。记录运动过程中每个小立方体的方位信息，生成动画文件。播放时，加载对应动画文件，生成动画信息，并应用到各个小立方体，形成堆叠效果。

## 关键词

动画、物理引擎、录制、3D、Unity。

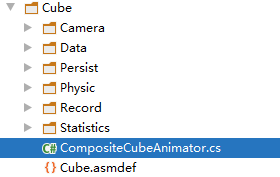
## 引言

小立方体堆叠成大立方体的视频展示了非常漂亮的立方体堆叠及散落效果，这个效果就是我们的研究目标。



## 正文

### 功能文件清单

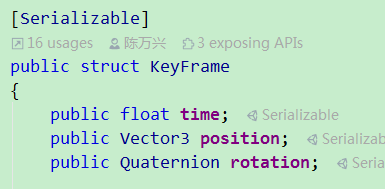


### 方案

运动动画采用记录关键帧的形式实现。关键帧包括时间（或帧id）以及方位信息。动画播放过程中，根据关键帧计算当前的运行状态。

### 关键帧

方位信息包括位置、定向以及缩放。为了节约内存使用以及目前的动画需求，可以忽略缩放（节约30%）。故目前存储的方位信息只包含位置和定向。方位信息附加时间戳形成动画关键帧信息。



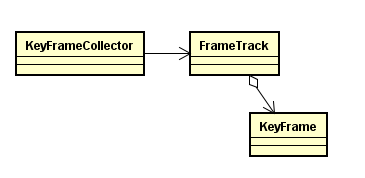
### 采样策略

动画片段存储的是离散的关键帧信息。如果采样时间为某一关键帧，这时返回对应关键帧信息即可。但是大部分情况下，采样时间会位于俩个关键帧之间，这时就需要根据前后关键帧信息进行线性插值计算得出动画信息。

不同的属性采用不同的插值策略。位置属性通过调用Vector3.LerpUnclamped计算。定向属性则通过Quaternion.SlerpUnclamped计算。

### 动画数据结构

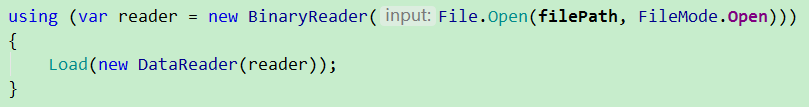
大立方体包含多个小立方体。每个小立方体对应一个动画片段。每个动画片段保存多个关键帧信息。对应类图如下：



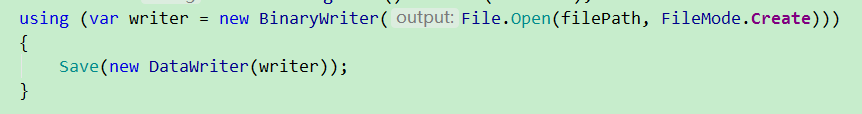
### 动画文件格式

为节约内存使用，提升序列化反序列化效率，动画信息的存储采用二进制文件的方式。

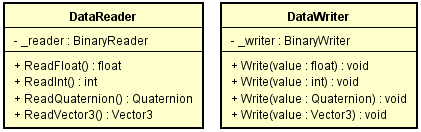
文件打开通过二进制读取流，代码如下：



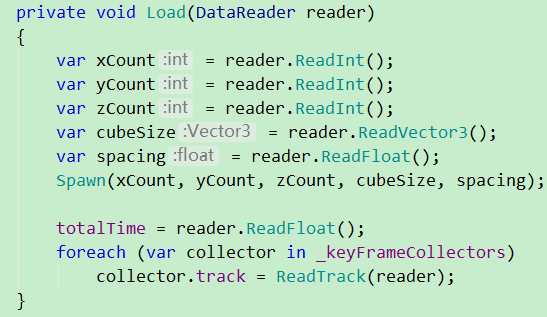
文件保存通过二进制写入流，代码如下：

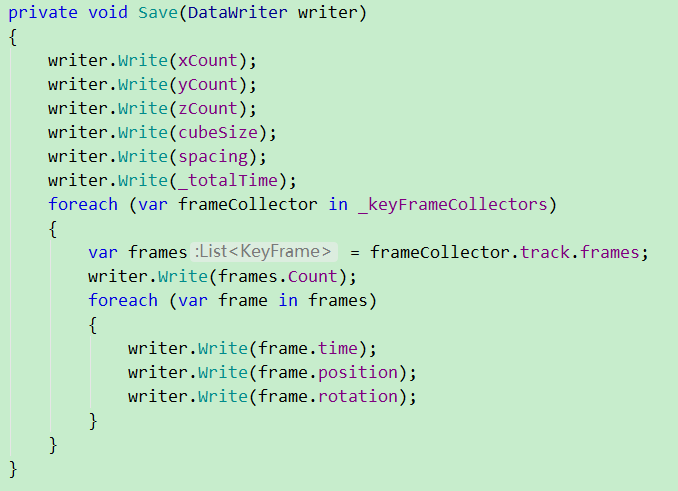


为方便使用，统一数据格式，抽象二进制数据读写类并封装通用的读写方法。



在保存加载回调中调用相应方法进行数据读写。





### 动画录制结束判断

一个小立方体对应一个动画片段。当这个立方体方位变更，则产生并记录关键帧。

如果所有的小立方体方位不再变更，则判定动画录制结束。

### 性能分析

实验数据：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **指标** | **URP渲染模式** | | **HDRP渲染模式** | |
| **每帧平均数据** | **性能消耗百分比数据** | **每帧平均数据** | **性能消耗百分比数据** |
| **CPU** | **1.378 ms（+ 0.202 ms）** | **4.08%（+0.4）** | **11.386 ms（+ 0.086 ms）** | **10.3%（+0.64）** |
| **GPU** | **0.292 ms（+ 0.026 ms）** | **34.24%（+10.38）** | **9.1596 ms（+ 0.039 ms）** | **60.5%（-1.62）** |
| **内存** | **-** | **142.6 MB（+ 5.1 MB）** | **-** | **2.20 GB（+ 0.01 GB）** |

## 结论

### 平台支持

PC、Android以及iOS

### 性能结论

立方体堆叠与散落动画播放CPU的耗时整体较小，从结构化数据分析，URP渲染模式下大量的时间仍花在CPU等待GPU垂直同步上

### 应用建议

1. 录制动画及播放动画分俩个场景进行配置。这样可以带来一些好处：
2. 配置方便，不用再录制以及播放来回修改物理配置
3. 若应用没有物理需求，可以剥离物理模块
4. 播放模块尽量使用组件的方式（播放组件支持编辑态）而不是API，这样能快速调整效果，并进行效果验证

## 参考

无