2023-12-30

计算机图形学 — 考试大作业

# 学号： 521021910760 姓名： 钱韦克

# 本次作业回答如下：

# 房间搭建

## 墙面和灯的模型建立：

### 对于所有墙面，建立Plane类。它的模型就是一个包含两个三角片四边形。在建立墙面的时候为每个点手动赋值坐标，法向量和材质坐标即可。而灯则借用了Ball类来实现，放在天花板上。最终，每个墙面的大小为2.0f，房间最中心的坐标为(0.0f,0.0f,0.0f),灯泡的位置在（0.0f,0.0f,0.5f）处。

### 

## 木纹材质，纯色材质和灯光材质

### 材质的显示已经在Mesh结构体的Draw函数中实现。因此只需导入材质后绑定材质坐标即可。

### 导入材质代码：

### 材质图片：

### 

### 而纯色的材质和灯光的材质仅仅由颜色rgb表示。例如：

## 根据灯光的光影效果实现

### 根据物体的材质，在着色器中可以获取到需要显示的颜色。接下来，通过读取向着色器中传入的Light结构体的参数：

### 

### 就可以在片段着色器中进行光照渲染的计算。

### 

### 分别计算环境光、漫反射光和镜面反射光。

### 环境光就是无视方向距离的默认光照。

### 漫反射光通过光线和法线的夹角来决定亮度。

### 

### 镜面反射光再进一步考虑视线方向来决定镜面反射的亮度。

### 

### 在计算出上面的冯氏光照后，再计算衰减：

### 

### 其中，系数选取了能照亮32的一次和二次项系数。

### 

### 最终合成出最终的光照。

### 由于光源是点光源，就没考虑聚光源的边缘模糊效应。

### 另外，对于发光体或粒子等物体，仅仅只显示物体的颜色本身，不会计算光照。

# 不倒翁系统建立

## 不倒翁模型读取和参数描述

### 模型读取

### 分别建立了Tumbler和Tumbler Cluster类来管理不倒翁。

### 模型读取的逻辑由Model类实现。只需提供模型路径即可。

### ii． 参数描述

### 对于一个不倒翁类，有以下参数来描述它的物理性质：

### 

### position：描述不倒翁重心的位置

### scale：描述不倒翁的大小

### norm类：描述不倒翁摇摆平面的法线性质

### axis类：描述不倒翁的摇摆性质

### self类：描述不倒翁的自转性质

### 上面三个的性质分别包括了角度、速度、能量损失程度和计算系数。

### 剩下的还有各种物理参数 ，例如质量和转动惯量等。以上将在描述物理拟真效果时详细介绍。

### 通过以上参数，便可以绘制出不同状态下的不倒翁模型。

### 

## 阴影贴图生成和绘制

### 创建深度贴图

### 

### 像别的材质一样，创建了一个深度阴影贴图。

### 渲染至深度贴图

### 从光的视角渲染每个需要生成阴影的物体，把对应的顶点变换到光空间，从而在渲染结束后生成深度缓冲。

### 因为模拟的是点光源而不是平行光，所以此处用的变换是投影矩阵。

### 

### 

### 渲染过程中需要将视口大小和阴影贴图大小改成一致。

### 渲染阴影

### 生成深度贴图后，就可以根据生成的深度贴图在着色器中渲染阴影。

### 具体就是需要在片段着色器的渲染过程中，通过储存到深度贴图中的深度值，我们就能找到最近点，用以决定片段是否在阴影中。

### 

### 

### 最后，根据阴影状态显示需要的颜色即可。

### 由于阴影贴图的分辨率不够，显示的阴影会有很明显的锯齿。因此，采用PCF方案来产生柔和的阴影边缘，增强显示效果。核心思想是从深度贴图中多次采样，每一次采样的纹理坐标都稍有不同。每个独立的样本可能在也可能不再阴影中。所有的次生结果接着结合在一起，进行平均化，就得到了柔和阴影。

## 鼠标点击的人机交互实现

### 当鼠标点击时，将鼠标点击的位置转换为真实空间里的方向向量，便能得到一条从摄像机位置出发的射线。

### 

### 在不倒翁中对射线进行碰撞检测，判断点击位置是否存在不倒翁，并且是在不倒翁的重心上部还是重心下部。

### 通过分析，不倒翁的形状可以大致抽象为上下两个半径1：2的球体加上腰部的一圈：

### 

### 而在射线检测中可以进一步简化，

### 

### 如果射线在上面这个大球体的范围内，则视作点击到了不倒翁的中心以上。否则，若射线在不倒翁的下面的球体的范围内，则视作点击到了不倒翁的重心以下。

### 若成功点击后，将鼠标模式改为GLFW\_DISABLED，并将所有鼠标的移动获取为坐标输入并交给Tumbler Cluster类处理，直到松开鼠标。

## 不倒翁的物理拟真效果实现

### 鼠标点击的平移

### 当鼠标点击到不倒翁的下部时，判断为不倒翁的平移运动。此时，鼠标在屏幕空间的xy位移分别对应到真实空间中x正z负两个轴的位移。将位移告诉tumbler类后，tumbler类便直接进行移动。在此时，对每一个预期坐标再进行一个额外的判定，判定是否会移出边界或者和其他不倒翁重叠。

### 

### 

### 鼠标点击的倾斜

### 同理，将鼠标在屏幕空间xy轴的移动交给不倒翁处理。

### 

### 实现逻辑为，将xy轴在操作期间的和转化为极坐标，并且强制限制它的长度来防止不倒翁过于倾斜。转换完之后，将对应的角度和长度转换给倾斜周的角度和倾斜度后，接下来就可以进行物理模拟了。

### 自转运动拟真

### 运动情况分析：

### 

### 即，不倒翁以自己的对称轴为重心，以selfRotate\_v的速率转动。在转动过程中，每时刻的能量损失速率为selfRotate\_EnergyLoss。通过能量方程来确定每时刻转动速度的变化情况。

### 受力情况分析：

### 

### 原本的真实情况是，在小球和不倒翁表面碰撞瞬间产生的摩擦力的冲量来导致小球运动速度的减慢和不倒翁旋转速度的改变。但是为了方便起见，我的实现中将这一步简化了。我所实现的方案是，当小球和不倒翁表面碰撞时，不倒翁在该表面位置处的速度瞬间和小球的速度在该表面切向量方向的分量一致，即碰撞瞬间使两者的速度相同。同时，也加入了一个自定义的系数self\_param，用来修正上述粗略计算过程中结果的单位规模问题。

### 经过以上的实现后，不倒翁便能在小球擦过其表面时产生自转，并且自转速度也会平滑的衰减。

### 对称轴旋转运动拟真

### 运动情况拟真：

### 

### 即，不倒翁摇摆平面法向量的旋转速度由normRotate\_v决定，同时这个旋转速度的衰减同样通过能量公式和能量衰减速率来确定，产生较为平滑和拟真的模拟效果。

### 受力情况分析。

### 

### 当不倒翁收到小球的冲击时，其法线旋转和摇摆幅度的改变只由在此瞬间收到的力决定。而对于这个碰撞情况而言，由于摩擦力仅仅影响自转的速度，因此其决定因素的只有沿着碰撞面法向量的作用力。通过动量定理：

### 加上小球运动情况的变化，可以得出小球的速率变化量，从而得出小球动量的变化量和力的冲量和大小。

### 当求得力后，根据角动量定理:

### 便可以计算得到不倒翁沿着y轴正轴方向的角动量，从而计算出不倒翁沿着y轴旋转的速度变化，也就是法线旋转的速度。和上一步一样，这一步也加入了自定系数norm\_param来改变计算结果的规模。

### 此外，由于此步骤中的旋转轴和上一步计算自转时对称轴位置的旋转轴不同，因此这里所采用的转动惯量normJ也和上面的内容不一样。实际上，这两步骤中的每一个物理参数都可以实时计算得出，但在做的时候为了方便还是使用了手动调参的方式来控制动画的效果。

### 摇摆运动拟真

### 运动情况拟真：

### 

### 摇摆部分的运动是最难实现拟真的部分。首先，和上面两个转动运动不一样的地方是，摇摆运动是一个简谐运动，因此需要额外的向量axisRotate\_a来记录运动过程中的加速度，才能正确描述运动的状态。而剩下的axisRotate\_v和axisAngle则仍然是描述摆动过程中的角度和瞬时角速度的变量。

### 同上面的一样，摆动同样需要计算衰减。从物理的真实角度而言，摆动过程的能量是由简谐运动的振幅决定的。对于不含衰减的简谐运动而言，它的能量就是

### 其中，J为不倒翁以旋转平面法向量为轴的转动惯量，k为该简谐运动的系数。两者同样也是方便起见通过自己定义来调整的，没有进行额外的物理学计算。

### 在不含衰减的简谐运动中，它的能量是不会改变的，因此它每个时刻的运动都是确定的，例如

### 但是当加入衰减后，它的振幅A也会随之衰减，从而导致加速度和速度的改变。我在计算过程中，尝试过同样用能量定理和自定义衰减来计算同一时刻前后的角速度，但就结果而言并不尽人意。即使计算能量的方法我尝试了很多种，包括试图忽略位移带来的影响，但都未能实现一个较好的效果。因此最终，还是采取了一个简单粗暴的办法：取得一个衰减系数，每次计算时仅仅将加速度乘以这个系数，速度本身不做影响——即减缓向外摆的速度，增加摆回中心的速度。通过这种方法，最后还是能以一个较为平滑的方式实现摆动幅度的过渡。

### 受力情况分析：

### 

### 对摆动的受力分析也比较复杂，也是从理论物理学角度变为自定义实现效果的偏路。

### 首先，将力按照不倒翁的变换矩阵的逆进行反向变换，然后得到力在不倒翁底部地板这个平面上的投影，作为影响不倒翁摆动的力。接下来，将原不倒翁的摆动抽象为该平面上一个向量。向量的角度即为不倒翁当前摆动偏移的方向，而向量的角度则为不倒翁当前摆动过程的能量，而不是幅度。接下来，再将影响力投影到平面上后，同样通过自定义的系数axis\_param将其转变为一个具有方向和长度的向量。将两个向量合成，得到的就是不倒翁在力的作用之后所期望的摆动向量。

### 接下来，反向将该摆动向量拆解，依次得到模长，能量，速度和加速度以及方向。此外，还要对速度增加一个限制，防止它的摆动过程中产生太大的角度。这个限制也是手动计算决定的。

### 得到新的摆动后，由于碰撞的该瞬间物体的位移不能产生瞬间变化，所以只能改变它的速度。这里我才用的方法是，将摆动的速度瞬间变为最终摆动向量在登长位移处的速度，并将法向量旋转速度增加一个从当前摆动角度到目标摆动角度乘以一个自定义系数的值，用来逐渐改变摆动的方向。

### 最终，通过以上所有步骤，不倒翁的物理模拟系统就完成了。由于涉及到许多未进行人为精确计算的物理变量，但是通过不断地反复修改自定义的各个变量，因此这个物理拟真虽然并没有完全按照现实物理拟真，但是也能产生一个较为顺滑和合理的物理运动效果。

### 

# 小球动画实现

## 多个小球的模型建立和生成逻辑

### 和多个不倒翁有一样，建立了Ball类和BallSystem类来管理所有球体。其中，Ball类建立网格体的逻辑如下：

### 

### 即枚举α和φ，得出球体球面上的多个点坐标的位置，再通过在这些点之间绘制三角片来完成球体的建模。同时，也为其赋予材质坐标，使纹理能够正常映射到该球面上。

### 当按下按键X（并不是要求中的按键S，因为按键S被我绑定到了摄像机的移动逻辑上去了）后，便会在位置（0，0，0）位置生成30个球体，并为每个球体赋予一个随机的初速度。

## 小球的材质动态实现和阴影

### 小球的阴影由和不倒翁一样，在生成阴影贴图的时候绘制生成深度缓冲即可。下面的火球也是一样。

### 

### 而材质实现的过程，则是在绘制小球的过程中向小球的着色器额外传递一个参数，告诉着色器这个小球需要以什么材质绘制。

### 

### 

### 由于每个小球的木地板材质已经在建立的时候赋予了，因此只需要改变着色器的这个type参数就可以很简单的实现材质的自定义显示。

## 小球的物理仿真

### 对于小球的自运动而言，它的速度只是一直由重力加速度改变。因此运动实现也很简单。

### 

### 而当小球产生碰撞时，只要得到碰撞地方的法向量，也很容易得到小球碰撞后的运动情况，即

### 

### 通过这个，便可以完全模拟出所有的小球运动状况了。需要注意的是，这里对小球的运动模拟假设是它的所有碰撞不会产生能量衰减，所以小球在地板弹跳的时候每次反弹的高度都会一样。此外，还增加了超过一定距离后删除小球的逻辑，防止其一直占用资源。

## 小球的碰撞检测系统

### 对小球的碰撞检测整合到了一个函数之中进行。

### 

### 这个函数将与各个碰撞定了一个优先级，只会先与优先级高的物体发生碰撞改变运动。检测的顺序先是不倒翁，再是墙面。

### 先从简单的说起。墙面碰撞的检测非常简单，由于所有墙面都是一个位置已知的平面，因此只要判断小球三个坐标距离房间大小的间隙是否小于它的半径即可。若产生碰撞，由于墙面的法向量处处一致，随便传一个法向量给小球就能模拟了。

### 

### 接下来是不倒翁的碰撞。

### 在先前射线检测的逻辑中，将不倒翁抽象成了上面的小球体和下面的大球体。

### 

### 但是在碰撞的模拟中，不能继续进行这么简单的抽象，否则无法正确计算不倒翁的运动模拟。

### 我在此处碰撞检测函数中采用的方法是，将不倒翁上下球体各取出半个出来，作为有效碰撞位置，而对它的身体部分，则将这一整个半锥形作为有效的碰撞体。

### 在产生碰撞后，首先检查是否和上下球体产生碰撞，即两个球心之间的距离是否小于半径之和。若是，再通过叉乘结果的正负性检查碰撞位置是在上半部还是下半部。

### 这一部分过程中，若成功检测到了有效碰撞，便再进一步计算对不倒翁自身和对小球产生的影响。对小球而言，法向量就是球心之间的向量。而对不倒翁而言，由于对其下半部分的力通过其重心，所以这部分力理论上而言只会产生位移，不会产生旋转方面的影响。但在计算过程中，我假设地板摩擦力无限大，因此并没有计算位移，所以也就不需要对不倒翁进行处理。而对其上半部分的碰撞，只要将碰撞信息交给不倒翁的物理拟真函数即可。

### 接下来，判断是否和不倒翁腰部这个几何体产生碰撞。

### 

### 首先，将立体的半锥体抽象化，仅仅在碰撞位置的共平面考虑，便成了一个2维的平面几何问题。先取得碰撞位置方向和垂直于不倒翁对称轴方向的夹角，再用三角函数得出从底面中心沿着这个夹角到半锥体外表面的距离。在将这个距离和碰撞位置到地面中心的距离和半径进行比较，便可以得知这个位置是否会和这个锥体进行碰撞。

### 但另外，由于这个几何体实际是个半锥体，因此需要对夹角进行一些限定。通过计算，来限制半锥体表面和碰撞方向射线交点到底部平面的距离要小于不倒翁高度即可。

### 

### 若检测到有效碰撞，则需要进一步计算碰撞位置法向量。由于这个锥体的角度是已定的，因此在这个平面上，法向量的角度也是处处一致的。剩下需要的，就是结合变换矩阵，将抽象平面上的法向量先旋转回不倒翁自身坐标系的位置，再进一步通过不倒翁变换矩阵旋回空间实际坐标系即可。

### 最后，将法向量分别给小球和不倒翁的物理模拟函数，便可以进行碰撞检测。

### 此外还有个小手段。在不倒翁身体的半锥体产生碰撞时，需要额外立刻改变小球的位置，不然由于不倒翁自己的旋转会导致小球反复检测到碰撞，反复被弹走又反复被吸回，最后黏在不倒翁面上一起旋转。

# 火球动画实现

## 单个粒子特效实现

### 一个粒子有如下参数：

### 

### mesh是渲染网格体，position和rotation是粒子运动的速度参数。

### 两个color是实现粒子渐变的起点终点颜色。

### size和sizeAttenuation是粒子大小的参数，后者决定粒子大小的衰减，来实现粒子消失的效果。

### lifetime和livedTime是粒子的寿命相关参数，同时用来决定当前显示的颜色。

### 后面4个则是用来计算粒子运动的物理学参数。

### 在实现一个粒子时，我的设计是将其设为一个正四面体。这个设计相比于正方体而言，只要用到4个三角片，和正方形的12个三角片而言少了3倍，也能提高渲染效率。

### 粒子在其生命周期中的计算函数如下：

### 

### 即，首先强制粒子不会穿透房间的墙壁（但会穿过不倒翁和小球）。然后，对齐进行速度的物理学拟真。这里的G不仅仅指的是重力向量，而是人工指定的粒子运动过程中所期望的外力带来的加速度。此外，还有一个需要考虑的点。由于粒子在考虑过程中，是密度质量很小的物体，所以除了重力之外和前面物体不同的是，它还应该额外计算一个空气阻力，即由系数f\_k决定的空气阻力来阻止粒子运动，它的大小和粒子运动速度的平方成正比。最后再通过生命周期相关函数对其大小和颜色做改动。以上就是一个粒子在其的一整个生命周期的执行逻辑。

### 而粒子渲染的着色器则只用显示颜色即可，不用考虑光照。

### 

## 粒子系统搭建

### 一个粒子系统有如下参数：

### 

### particles是shared\_ptr类型的vector数组，用来管理这个粒子系统中所有的粒子。shared\_ptr类型会自动化释放内存，因此无需担心内存泄漏问题。

### centerPos是粒子系统的中心位置。

### emitSpeed是粒子系统发射粒子的速率。此处的速率指的不是粒子运动的速率，而是单位时间发射粒子的个数。

### consist用来描述该粒子系统是一直产生粒子的类型还是瞬间产生粒子的类型。

### systemV用来描述系统的运动速度。

### 接下来的是用来赋予其生成粒子的参数，包括颜色，生命周期，速度，大小，外力。

### 再后面是生成粒子过程使用的参数。其中，oriented用来描述生成的粒子是否是沿某个方向的。若没有指定，则是在球形范围内随机生成，否则就是在以指定方向为对称轴的半球体范围内生成。

### 

### 粒子系统生成粒子和运动状态更新的逻辑和之前提及的没有什么区别，仅仅是将参数传递给目标类而已。需要注意一点的是，在粒子系统中运动的粒子，会由于粒子系统自身的速度而产生自己的惯性，所以生成的速度要额外加上系统的速度。

### 而当需要激活粒子系统时，需要为其传递参数，它便可以在每个周期内或瞬间按emitSpeed随机生成粒子了。

### 

### 例如，我另外实现了一个专门利用我的这个粒子系统来生成粒子动画的类。

### 

### 它实现的灰烬和火花效果如上显示。只要递给粒子系统参数即可，不同的参数就能实现不同的期望效果。

## 火球的构建和阴影

### 火球的阴影构建如小球一样，这里不多讲述。

### 而火球的构建也和小球基本一致。唯一一个不一样的是，火球自身还包含了额外一个自己的粒子系统，用来显示它的火焰。

### 。

### 另外，火球的运动和小球不一样，它一旦被发射，就会沿既定的方向按照自定的速度运动，不会收到外力影响来改变运动状态。

### 

### 同样，它的粒子系统的构建也只是传递了参数。在这个参数中，除了系统的速度设为小球的速度外，将外力的加速度设为了0。

### 在渲染过程中，也只要将中心的小球按照发光体着色器渲染，粒子系统按照粒子着色器渲染即可。

### 而生成火球的方式和之前检测射线中的步骤一致，即将鼠标位置转换为空间射线，然后将火球按照这个射线发射。

### 

## 火球的碰撞检测和效果生成

### 火球的碰撞同样以额外的函数进行整合。

### 在这里，它额外计算了和小球的碰撞，并将小球碰撞的优先级设为了最高。检测小球碰撞的逻辑很简单，只要判断两者球心之间的距离和两者的半径之和。而剩下的对不倒翁和墙面的碰撞检测和之前小球部分相同。

### 产生碰撞后，对于得到的法向量，它不是拿来更新火球的运动状态，而是传递给动画系统，让其生成有方向的瞬间粒子系统，用来显示特效。对于碰撞表面，沿物体表面的法向量生成火花特效，若是小球便额外以反方向生成灰烬特效。

## 火球的光影效果

### 由于火球也算是一个发光体，因此在它生成的时候也应该照亮其他物体。所以，更新所有渲染光照的着色器，为其计算复合光照：

### 计算原理和之前天花板灯的计算原理一直，并在最后显示结果的时候将两者的光照相加即可。

### 而在主函数中，每个周期更新渲染着色器中火球相关的参数。

### 这个用来告诉渲染器火球的状态和位置。

### 另外在最开始初始化的时候，也要告诉每个着色器火球的光照效果。

### 

### 在这里，将火球光照的环境光设为了0，镜面反射光设的很强，而散射光系数设为了0.3f。

### 经过以上步骤，便可以在火球生成时在房间观察到被火球照明的效果已经墙壁上的镜面反射效果。