过程安排 Timeline

事项	截止时间
公布选题	2022.4.1（第7周周五）
组队、选题完成（期间请与mentor充分沟通选题内容和要求、确定适合的题目；自拟题目小组也请在此期间和老师助教沟通好）	2022.4.8（第8周周五）
Mentor为选题小组进行课题所需背景知识的tutorial和答疑	第9周周日(2022.4.17)之前
Mentor和选题小组每两周进行一次集中线上讨论（其他时间可单独和Mentor讨论）	第11周周日(2022.5.1)、第13周周日(2022.5.15)、第15周周日(2022.5.29)之前
答辩	第16~17周（时间待定）
提交最终报告	2022.6.12（第17周周日）

选题要求

1. 自由组队，每组成员不超过三人；
2. 每个小组可在我们给出的选题中自由选择；
3. 如有小组想自己寻找课题，可单独与老师或助教沟通；
4. 4月8日前确定小组和选题，请各小组通过下方问卷提交（扫描下方二维码进入问卷）

答辩形式 (第16~17周 (时间待定))

- 7分钟小组ppt展示+3分钟提问。
- ppt展示包含：选题背景、研究方法、实验结果、创新性、总结。
- 答辩评分标准：

打分项	分值
项目完成质量	40'
方法创新性	20'
陈述清晰度	20'
回答问题情况	20'

最终提交形式 (2022.6.12第17周周日)

- 4-12页大作业报告（中文，无固定模板，格式word、pdf均可）
- 报告包含：简介与意义，相关工作，研究内容与方法，实验结果与分析，特色与创新，总结。
- canvas上提交：1）报告，2）源码及使用文档，3）效果展示录屏，4）小组分工。分为4个文件夹、打包上传。
- 最终提交评分标准：

打分项	分值
课题背景及相关工作	20'
项目完成度	40'
创新性	20'
结果展示质量	20’

可选题目概览

- 1、实时阴影（mentor：王玥）
- 2、基于笔画序列的人脸风格化（mentor：胡腾）
- 3、灰度图像上色（mentor：王玥）
- 4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）
- 5、基于 Jittor 框架的艺术风格人脸生成（mentor：王玥）
- 6、图像中物体表面材质反射率估算（mentor：徐添辰）
- 7、大量物体实时碰撞检测（mentor：徐添辰）

1、实时阴影（mentor：王玥）

- 课题任务：
  - 1）硬阴影：实现Shadow Map方法。
  - 2）软阴影：实现PCF和PCSS。
- 本项目提供基于WebGL的代码框架[附件链接]，主要需要完成对phongFragment.glsl 中函数的完善
- 课题要求：

1. 实现Shadow Map 算法，能够看到图像中两个模型的硬阴影。
2. 实现PCF 算法，能够看到图像中模型的阴影在边缘处的模糊。
3. 实现PCSS 算法，能够看到图像中模型的阴影在遮挡物和阴影之间距离近的地方阴影偏硬，在距离远的地方阴影偏软。
4. （不做要求、加分项）多光源下的shadow map，和动态物体的场景。

- WebGL学习资料：https://developer.mozilla.org/zh-CN/docs/Web/API/WebGL_API/Tutorial
- GLSL学习资料：https://learnopengl-cn.readthedocs.io/zh/latest/01%20Getting%20started/05%20Shaders/

1、实时阴影（mentor：王玥）

- 说明：
  - 需要使用WebGL，硬件平台无要求
  - 更多提示请见附件[附件链接]
  - 本课题参考Games202作业1

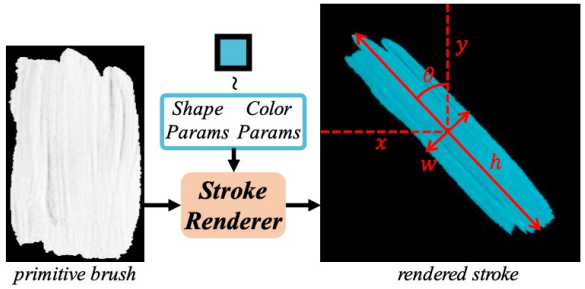
- Mentor：王玥 e-mail：imwangyue@sjtu.edu.cn

2、基于笔画序列的人脸风格化（mentor：胡腾）

- 课题任务：以序列化笔画的方式完成对人脸图像的风格化。
- 任务介绍——什么是序列化笔画？
  - 本实验以笔画作为基本的图像组成单位而非像素，模仿人类画画的方式，预测由第一个笔画到最后一个笔画构成的笔画序列，然后根据预测的笔画渲染出由笔画序列构成的图像。
  - 其中笔画由笔画参数化模型描述，即通过笔画的位置、形状、颜色等参数描述一个笔画（见右图）
- 课题要求：
  1. **从下页3篇参考论文中任选一个**，完成序列化笔画的人脸风格化实验：将参考论文中方法应用于人脸照片、实现艺术风格化。
  2. 针对人脸照片的特点对已有算法进行改进。
- 测试数据：CelebA-HQ数据集中的人脸照片

由一个个笔画构成图像的过程(gif)

笔画的参数化模型示意图
(位置xy,长宽hw,旋转角度θ,
颜色rgb)



2、基于笔画序列的人脸风格化（mentor：胡腾）

- 参考论文：

1. Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes. CVPR 21

https://github.com/CompVis/brushstroke-parameterized-style-transfer 支持CPU、GPU(最低显存：12GB), GPU平台可用Google Colab或Kaggle。

2. Stylized Neural Painting. CVPR 21

https://github.com/jiupinjia/stylized-neural-painting 支持CPU、GPU(最低显存：8GB)

3. Paint Transformer: Feed Forward Neural Painting with Stroke Prediction. ICCV 21

https://github.com/Huage001/PaintTransformer 需要GPU(最低显存：6GB)

注：第一篇论文为基于笔画的风格化，第二篇为基于笔画的重建与风格化，两者均可用CPU；第三篇仅有基于笔画的重建，如需做风格化需要一定修改

- Mentor：胡腾　　e-mail：p.l.a.y.e.r@sjtu.edu.cn

3、灰度图像上色（mentor：王玥）

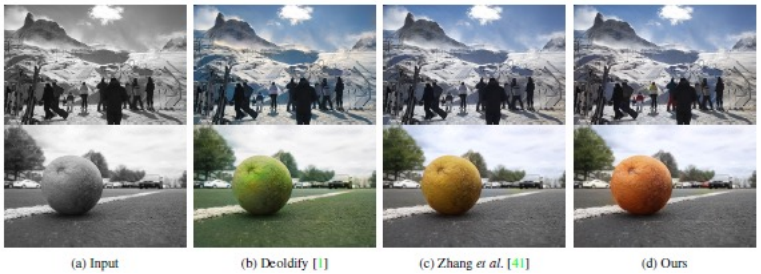
- 课题任务：完成灰度图像上色实验
- 任务介绍—— 灰度图像上色
  - 将灰度图像转换为彩色图像。（输出的彩色图像不需要同输入灰度图像对应的原始图像颜色完全相同，只需恢复的彩色图像尽可能自然）利用深度学习模型学到的先验知识完成上色。
- 课题要求：

1. **从相关工作3种方法中任选一个**，在ImageNet数据集完成测试实验（参考论文实验设置，可使用预训练模型）
2. 对已有上色方法进行改进

Deoldify上色效果

3、灰度图像上色（mentor：王玥）

- 相关工作参考
  1. DeOldify 老照片修复的开源软件
     - https://github.com/jantic/DeOldify
  2. Instance-aware Image Colorization (CVPR 2020) 结合图像分割方法的图像上色
     - https://ericsujw.github.io/InstColorization
  3. Colorization Transformer (ICLR 2021) 结合Transformer网络结构的图像上色
     - https://github.com/google-research/google-research/tree/master/coltran
- 硬件资源
  - 需要GPU支持，推荐显存12G以上，GPU平台可用Google Colab或Kaggle。
- Mentor：王玥 e-mail：imwangyue@sjtu.edu.cn

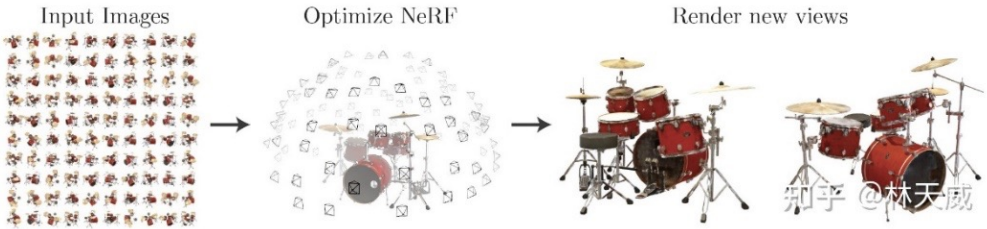


Instance-aware Image Colorization


Colorization Transformer

4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）

- 课题任务：使用 Jittor 框架复现 NeRF 相关论文
- 任务介绍
  - Jittor 框架
    - 计图（Jittor）：一个完全基于动态编译（Just-in-time）,内部使用创新的元算子和统一计算图的深度学习框架
    - https://cg.cs.tsinghua.edu.cn/jittor/
  - NeRF（神经辐射场）



    - 用神经网络，从一组2D图片隐式地学习一个静态的 3D 场景
    - 使用 volume rendering 进行渲染
    - Jittor框架对NeRF进行了实现[link]
- 课题要求：
  1. **从以下3篇参考论文中任选一个**，使用 Jittor 框架复现，比较Jittor和Pytorch中的运行速度
  2. 从3个基于 NeRF 的子任务中任选其一, 对已有算法进行改进

4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）

- 注：
  - Nerf论文：NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV 2020. Project page: https://www.matthewtancik.com/nerf
  - 基于Pytorch的NeRF实现: https://github.com/yenchenlin/nerf-pytorch
  - 基于Jittor的NeRF实现: https://github.com/Jittor/jrender#进阶教程3：NERF
  - Pytorch 转Jittor工具：https://cg.cs.tsinghua.edu.cn/jittor/tutorial/2020-5-2-16-43-pytorchconvert/

4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）

- 参考论文1，Faster Inference:
  - KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. ICCV 2021.
    https://openaccess.thecvf.com/content/ICCV2021/html/Reiser_KiloNeRF_Speeding_Up_Neural_Radiance_Fields_With_Thousands_of_Tiny_ICCV_2021_paper.html
  - 经典的神经辐射场方法在渲染图像时需要非常大的计算量，即使在高端硬件上仍难以实现实时交互性。本任务试图探索加速 NeRF 的渲染过程，使其能够在消费级 GPU 上以更快的速度渲染高保真的图像.
  - 代码（Pytorch）：https://github.com/creiser/kilonerf
  - 硬件要求：GPU: >= NVIDIA GTX 1080 Ti with >= 460.73.01 driver

4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）

- 参考论文2，Deformable:
  - D-NeRF: Neural Radiance Fields for Dynamic Scenes. CVPR 2021. https://openaccess.thecvf.com/content/CVPR2021/html/Pumarola_D-NeRF_Neural_Radiance_Fields_for_Dynamic_Scenes_CVPR_2021_paper.html
  - 经典的神经辐射场方法只适用于静态场景,。本任务探索将神经辐射场扩展到动态场景，允许在刚性和非刚性运动下重建和渲染物体的新图像.
  - 代码（Pytorch）：https://github.com/albertpumarola/D-NeRF
  - 硬件要求：GPU >= NVIDIA GTX 1080, about 2 days training

4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）

- 参考论文3，Few-shot:
  - pixelNeRF: Neural Radiance Fields from One or Few Images. CVPR 2021.
  - https://openaccess.thecvf.com/content/CVPR2021/html/Yu_pixelNeRF_Neural_Radiance_Fields_From_One_or_Few_Images_CVPR_2021_paper.html
  - 经典的神经辐射场的方法涉及对每个场景独立进行优化，需要许多校准的视图和巨大的计算成本。本任务试图探索在多个场景中训练模型以学习场景先验知识，使其能够从稀疏的视图集(少至单张)进行新的视图合成。
  - 代码（Pytorch）：https://github.com/sxyu/pixel-nerf
  - 硬件要求：NVIDIA GPU，drivers supporting CUDA 10.2

4、基于 Jittor 框架的 NeRF 研究（mentor：舒子曦、陈昂）

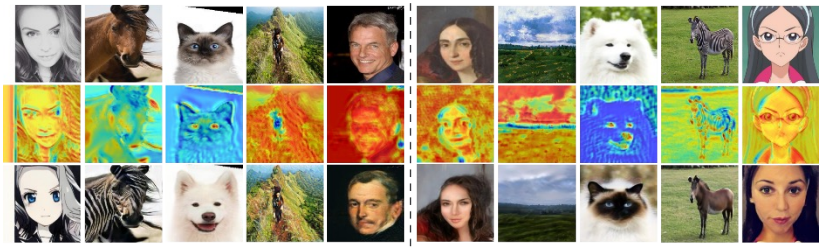
- Mentor:
  - 舒子曦　　e-mail：zixi.shu@sjtu.edu.cn
  - 陈　昂　　e-mail：ca.chen@sjtu.edu.cn

5、基于 Jittor 框架的艺术风格人脸生成（mentor：王玥）

- 课题任务：使用Jittor平台实现艺术风格人脸生成。
- 任务介绍——艺术风格人脸生成
  - 从真实人脸照片生成艺术风格的人脸图像（如梵高油画风格、卡通/漫画风格、素描等）。利用深度学习方法、基于Image-to-Image Translation框架进行艺术风格人脸的生成。
- 课题要求：

1. **从3个相关工作中任选一个**进行Jittor平台的复现，比较Jittor和Pytorch中的运行速率
2. 在人脸照片上进行测试，并针对人脸风格化进行改进

- Pytorch 转Jittor工具：
  https://cg.cs.tsinghua.edu.cn/jittor/tutorial/2020-5-2-16-43-pytorchconvert/
- 数据集和相关project总结：
  https://docs.qq.com/doc/DWVd0SndpQ2ZIVHV3



U-GAT-IT

AnimeGANv2

5、基于 Jittor 框架的艺术风格人脸生成（mentor：王玥）

- 相关工作：
  1. U-GAT-IT (ICLR 2020) 无监督图像转换方法

     Pytorch实现版本：https://github.com/znxlwm/UGATIT-pytorch
  2. AnimeGANv2 (ISICA 2019) photo -> anime 方法

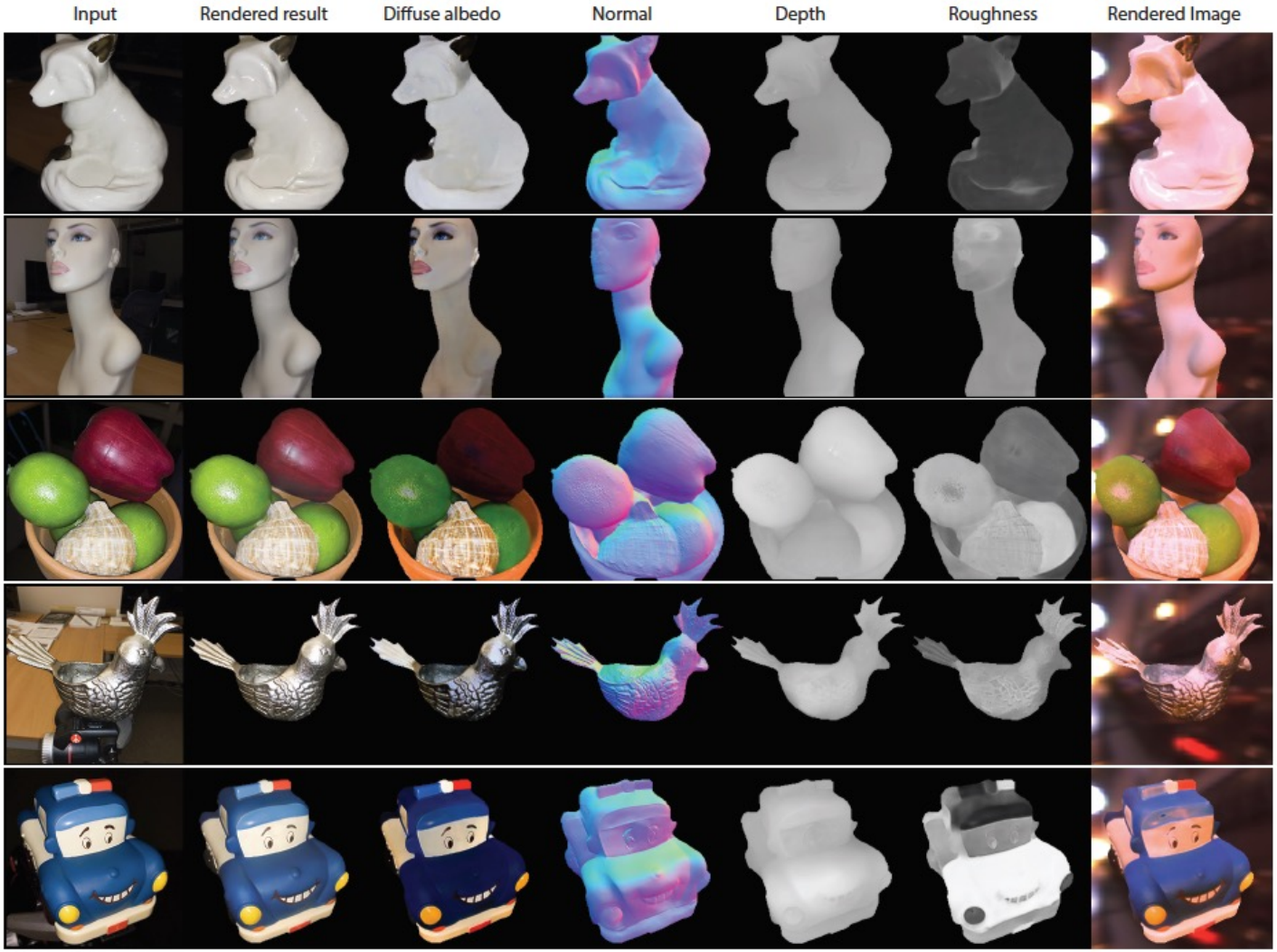
     Pytorch实现版本：https://github.com/bryandlee/animegan2-pytorch
  3. CartoonGAN (CVPR 2018) 图像卡通化

     Pytorch实现版本：https://github.com/Yijunmaverick/CartoonGAN-Test-Pytorch-Torch
- 硬件要求：
  - 需要GPU支持，推荐显存12G以上，GPU平台可用Google Colab或Kaggle。
- Mentor：王玥 e-mail：imwangyue@sjtu.edu.cn

6、图像中物体表面材质反射率估算（mentor：徐添辰）

- 问题
  - 估算图像中物体表面材质的反射率。
- 内容：AR中环境与物体光照一致性是一个重要问题。例如在一个视频中，使用AR技术将虚拟三维物体渲染到该环境中，如果地面反射率很高，虚拟物体将在地面呈现倒影，否则这个AR的表现就缺乏真实感。估算图像中地面材质的反射率（或者基于物理渲染PBR的参数：粗糙度、金属度）。
- 要求：从一张图像中估计物体表面材质的反射率，并绘制简单的动态三维物体在图像中呈现反射影像。
- 图像的地面不同区域有反射率差异，反射率难以手动标记。其他辅助标记可以手动添加。开发工具不限。
- 输入可以是图像或视频。基本输出为地面区域每个像素标记反射率，演示输出为简单三维物体在图像地面上呈动态反射影像的动画。

6、图像中物体表面材质反射率估算（mentor：徐添辰）



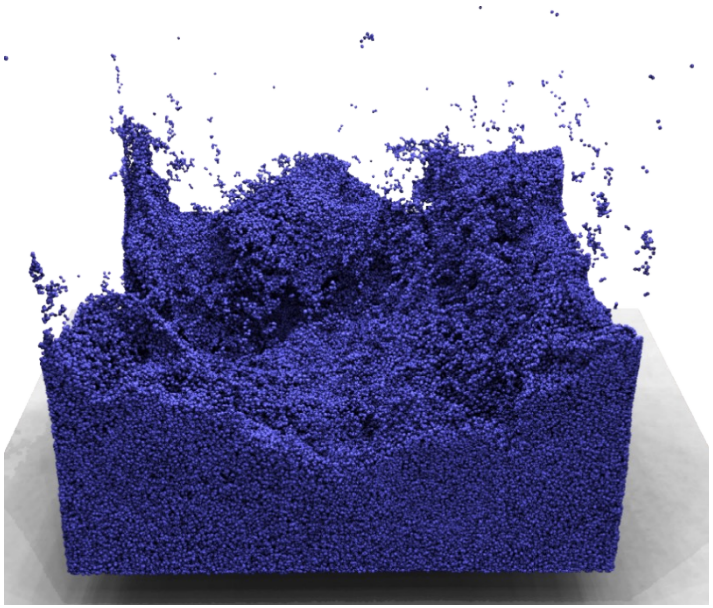
6、图像中物体表面材质反射率估算（mentor：徐添辰）

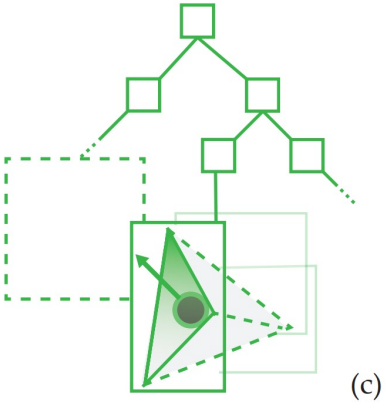
- 相关参考
  - Learning to Reconstruct Shape and Spatially-Varying Reflectance from a Single Image. SIGGRAPH Asia 2018.
  - 论文： https://par.nsf.gov/servlets/purl/10108150
  - 代码（Pytorch）： https://github.com/lzqsd/SingleImageShapeAndSVBRDF
  - Project page： https://cseweb.ucsd.edu/~viscomp/projects/SIGA18ShapeSVBRDF/
  - 硬件要求：基于深度学习需要相关GPU支持，也可以探索不使用深度学习的方法
- Mentor：徐添辰 xutc@ios.ac.cn tianchenx@outlook.com

7、大量物体实时碰撞检测（mentor：徐添辰）

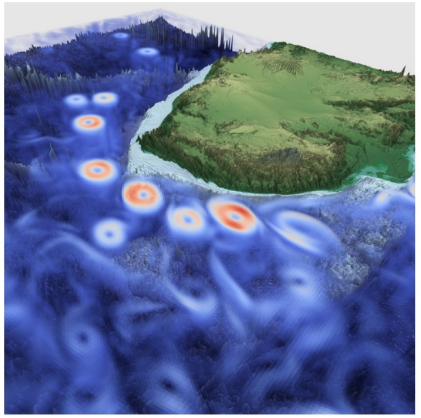
- 问题
  - 实现实时物与物之间碰撞检测。物体可用球表示，数量要求在规定值以上。
- 内容：VR作为实时交互系统，其中的图形物体碰撞检测是个经典问题，但当物体数目庞大时，碰撞检测算法已经很耗时。随着硬件发展，硬件也为此功能提供更多实现方案的选择。
- 要求：可以用UE4/5或Unity实现，也可以用API（OpenGL/DirectX/Vulkan/WebGL）实现。可以选择使用GPU实现，甚至也可以使用光线追踪实现。使用非光线追踪方法时，物体数量要求5万以上，帧率达到30/60fps（根据运行机器配置情况）以上。
- 输入：规定数目以上的三维球（可过程式生成）
- 输出：大量球在方盒范围内倾倒、坠落，以及互相碰撞的动画。

7、大量物体实时碰撞检测（mentor：徐添辰）





(c)



7、大量物体实时碰撞检测（mentor：徐添辰）

- 相关参考
  - https://developer.nvidia.com/gpugems/gpugems2/part-v-image-oriented-computing/chapter-37-octree-textures-gpu
  - https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-32-broad-phase-collision-detection-cuda
  - https://www.sci.utah.edu/~wald/Publications/2019/rtxPointQueries/rtxPointQueries.pdf
  - http://sci.utah.edu/~will/papers/rtx-points-tvcg20.pdf
  - 硬件要求：一般需要GPU支持，最低AMD Polaris架构400系/NVIDIA Maxwell架构900系，或者intel同等功能；如选择光线追踪实现，AMD RDNA2架构6000系/NVIDIA Turing架构2000系（此配置以下光线追踪使用DXR fallback）。
- Mentor：徐添辰 xutc@ios.ac.cn tianchenx@outlook.com