选题参考

下面给出的选题仅供参考，你完全可以尝试这之外的选题，比如在我们 lab 的 base code 上实现一个游戏。我们给每个选题标记了上手难度，⋆ 越多难度越大。但每个题目的难度上限都是非常高的。最终我们会综合 lab 选题的难度和同学们的完成度进行打分。大多数题目我们希望同学们在我们提供的 lab 基础代码上实现，这样能避免代码抄袭。特别是针对 lab 实现算法的改进，我们要求必须在 lab 代码基础上实现。如果不使用我们的 lab 基础代码，请提供完整的代码编译过程，并在代码中尽可能注释说明实现逻辑。

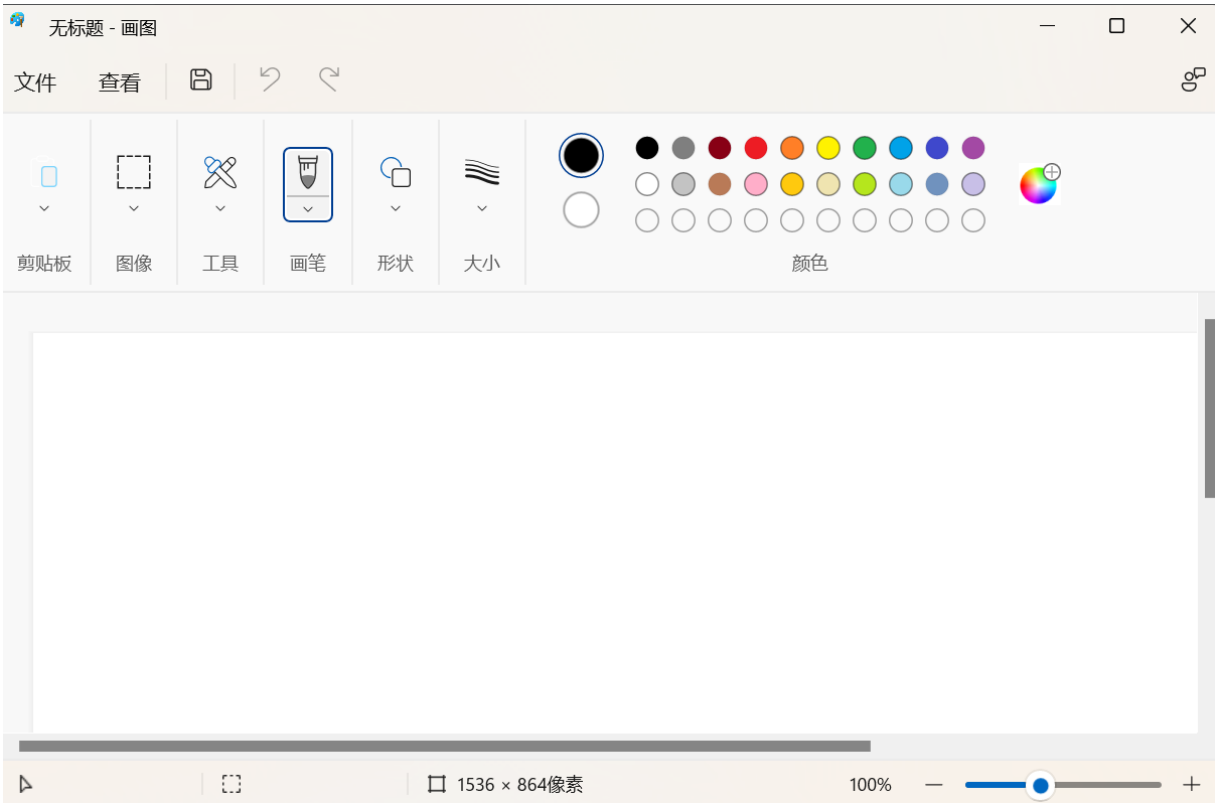
这里是一些可能用到的链接：

1. https://kesen.realtimerendering.com/：历年来 SIGGRAPH、SCA 等图形学会议文章的合集
2. https://dl.acm.org/: ACM 官方的论文库，可以用来查文章引用与被引
3. https://github.com/：寻找相关的开源代码，可以借鉴但是禁止直接复制粘贴

A. 2D drawing

1. Drawing Software (⋆)

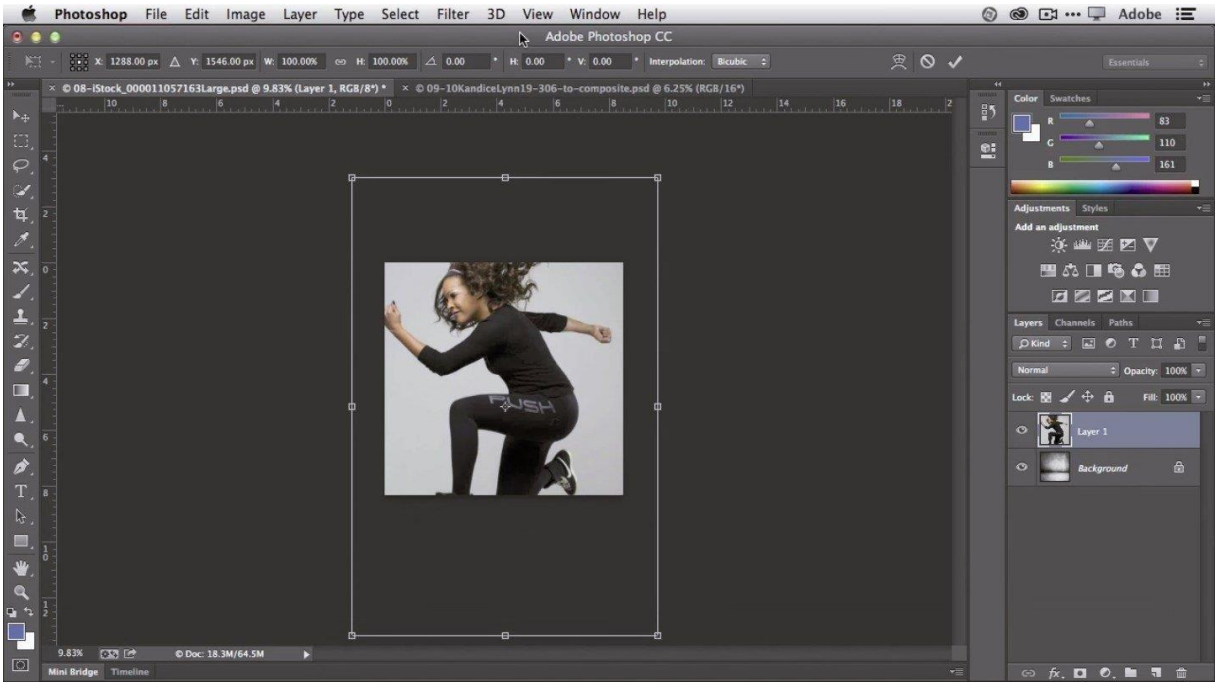
在 Lab1 中我们实现了画直线、曲线、三角形的各种算法，这些算法是实现一个绘图板的基础，比如 Windows 操作系统中自带的绘图软件就包含了这些功能：



在这个任务中，你可以在 lab 代码的基础上为这些算法设计更好用的 UI 交互界面，就像专业的绘图软件一样。并且，你也可以实现更复杂的功能，比如不同形状的笔刷、半透明效果等等。请发挥想象力用自己设计的画图软件创造出自己的艺术作品！

2. Image Editting Software (★)

在 Lab1 中我们实现了一些图像编辑功能，比如高斯模糊、边缘提取、图像补全等等，这些功能都包含在专业的图像处理软件中，比如 Photoshop：



在这个任务中，你可以观察学习现代图像处理软件中都包含了哪些功能，以及这些功能背后的原理，然后在 lab 代码的基础上实现一个包含尽可能多功能的图像编辑软件。

3. Practical Pigment Mixing for Digital Painting（★★）

https://github.com/scrtwpns/mixbox

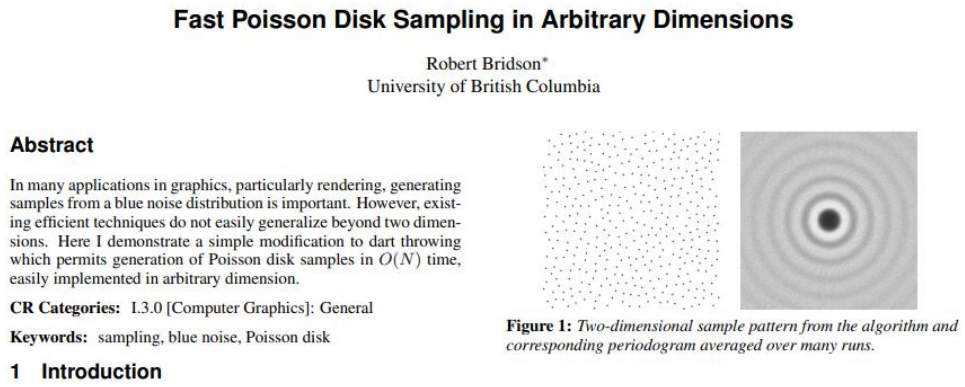
Practical Pigment Mixing for Digital Painting

ŠÁRKA SOCHOROVÁ and ONDŘEJ JAMRIŠKA,
Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic and Secret Weapons, Czech Republic

这是一篇 SIGGRAPH 2021 上的论文。作者注意到在现在的画图软件中，颜色的混合并没有按照颜料混合的规律。比如上图中，蓝色和黄色的颜料混合应该出现绿色，但是各种专业的绘图软件都没有混合出绿色。作者提出的了一种简单的方法在 RGB 的颜色空间中按照颜料的方式混合颜色的方法，并提供了开源代码。你可以首先阅读论文和对应的讲解视频理解算法的原理，然后在 lab 的代码基础上复现出来，并提供工具来使用这个混合方法创造画作！

4. Fast Poisson Disk Sampling in Arbitrary Dimensions (★ ★)

https://www.cs.ubc.ca/~rbridson/docs/bridson-siggraph07-poissondisk.pdf



Fast Poisson Disk Sampling in Arbitrary Dimensions

Robert Bridson*
University of British Columbia

Abstract

In many applications in graphics, particularly rendering, generating samples from a blue noise distribution is important. However, existing efficient techniques do not easily generalize beyond two dimensions. Here I demonstrate a simple modification to dart throwing which permits generation of Poisson disk samples in $O(N)$ time, easily implemented in arbitrary dimension.

CR Categories: I.3.0 [Computer Graphics]: General

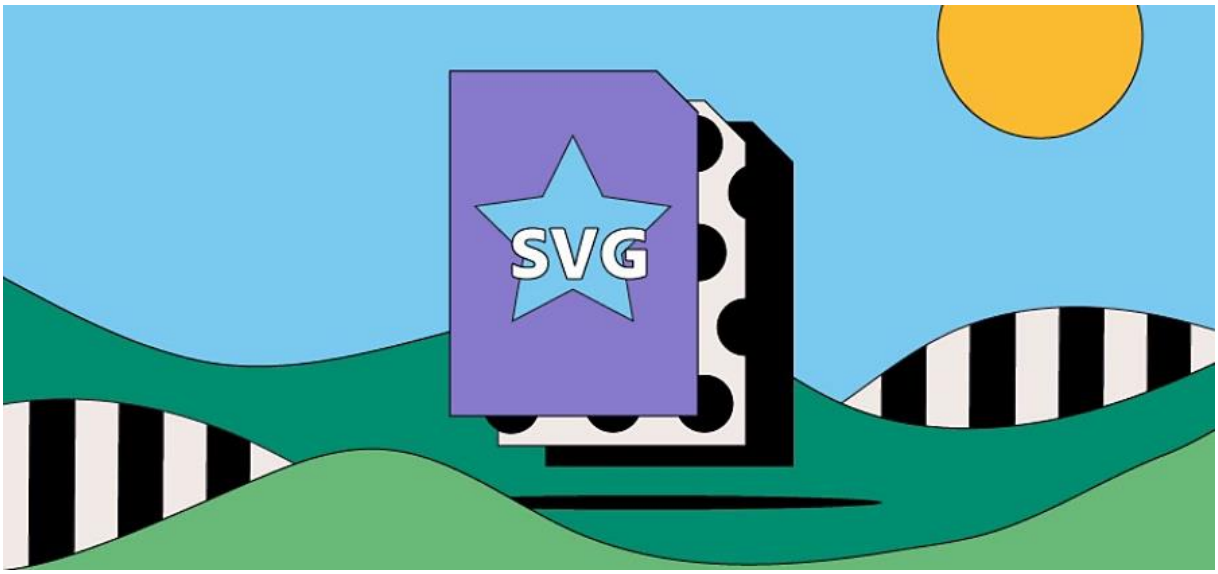
Keywords: sampling, blue noise, Poisson disk

1 Introduction

Figure 1: *Two-dimensional sample pattern from the algorithm and corresponding periodogram averaged over many runs.*

这是一篇 SIGGRAPH 2007 上的论文，正文加上引用一共只有一页。作者在这篇 paper 里提出了一种快速生成蓝噪声采样的方式。我们在 lab1 的 dithering 中使用了蓝噪声贴图，而蓝噪声在图形学中的应用远不止此。在这个任务中，你需要理解并复现论文中的算法，并通过实验或者理论分析证明算法的高效性，然后将算法应用到各种相关的下游任务中。

5. SVG Renderer (★ ★ ★)



SVG 格式是互联网上常用的矢量图格式，经常用于各种图标。不用于 png 等位图格式，SVG 里面直接描述组成图形的点、线、面，然后通过渲染器渲染成屏幕可以显示的位图。你需要首

先理解 SVG 文件的格式，然后解析其中的元素分别渲染，再最终组合在一起。

B. Geometry

1. Improved Geometry Processing (★)

在 lab2 中我们实现了很多经典的几何处理的算法，但这些算法并不是完美的，每个算法都有很多改进的工作。你可以调研这些改进工作，然后复现其中的一篇或者几篇，展示改进的效果。一些 survey 的 paper 和几何处理课程会是很好的入手点。

2. Point Cloud Surface Reconstruction (★★)

https://vgc.poly.edu/~csilva/papers/tvcg99.pdf

The ball-pivoting algorithm for surface reconstruction

The Ball-Pivoting Algorithm for Surface Reconstruction

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier,
Cláudio Silva, *Member, IEEE*, and Gabriel Taubin, *Senior Member, IEEE*

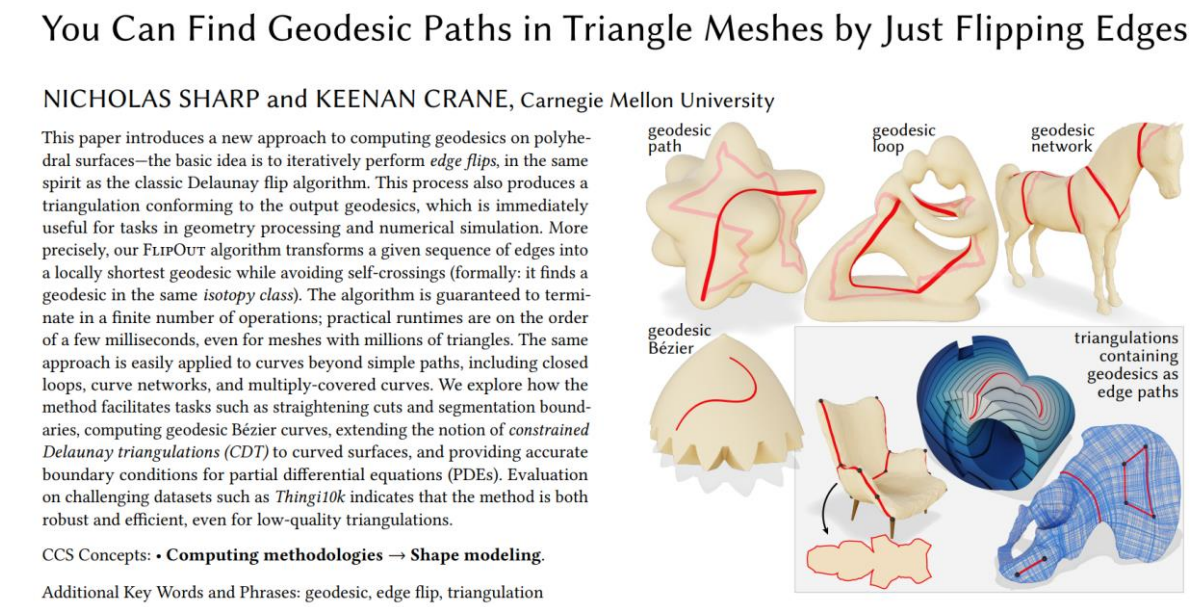
Abstract—The Ball-Pivoting Algorithm (BPA) computes a triangle mesh interpolating a given point cloud. Typically, the points are surface samples acquired with multiple range scans of an object. The principle of the BPA is very simple: Three points form a triangle if a ball of a user-specified radius ρ touches them without containing any other point. Starting with a seed triangle, the ball pivots around an edge (i.e., it revolves around the edge while keeping in contact with the edge's endpoints) until it touches another point, forming another triangle. The process continues until all reachable edges have been tried, and then starts from another seed triangle, until all points have been considered. The process can then be repeated with a ball of larger radius to handle uneven sampling densities. We applied the BPA to datasets of millions of points representing actual scans of complex 3D objects. The relatively small amount of memory required by the BPA, its time efficiency, and the quality of the results obtained compare favorably with existing techniques.

Index Terms—3D scanning, shape reconstruction, point cloud, range image.

从点云中重建出表面是一个经典的问题，1999 年在 TVCG 上提出的 Ball-Pivoting algorithm 是其中一个经典的算法。你可以阅读并在 lab 的代码基础上复现这篇论文的效果，或者也可以实现别的点云表面重建算法。

3. You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges (★ ★ ★)

https://nmwsharp.com/media/papers/flip-geodesics/flip_geodesics.pdf



You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges

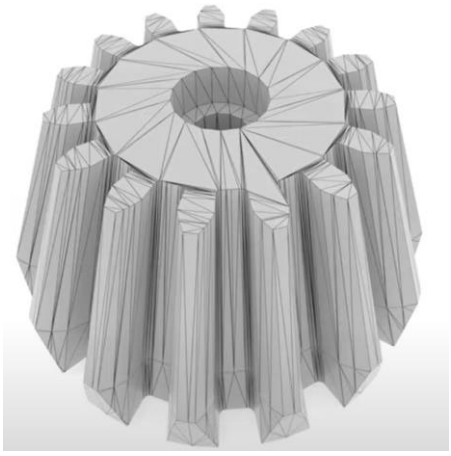
NICHOLAS SHARP and KEENAN CRANE, Carnegie Mellon University

This paper introduces a new approach to computing geodesics on polyhedral surfaces—the basic idea is to iteratively perform *edge flips*, in the same spirit as the classic Delaunay flip algorithm. This process also produces a triangulation conforming to the output geodesics, which is immediately useful for tasks in geometry processing and numerical simulation. More precisely, our FlipOut algorithm transforms a given sequence of edges into a locally shortest geodesic while avoiding self-crossings (formally: it finds a geodesic in the same *isotopy class*). The algorithm is guaranteed to terminate in a finite number of operations; practical runtimes are on the order of a few milliseconds, even for meshes with millions of triangles. The same approach is easily applied to curves beyond simple paths, including closed loops, curve networks, and multiply-covered curves. We explore how the method facilitates tasks such as straightening cuts and segmentation boundaries, computing geodesic Bézier curves, extending the notion of *constrained Delaunay triangulations (CDT)* to curved surfaces, and providing accurate boundary conditions for partial differential equations (PDEs). Evaluation on challenging datasets such as *Thingi10k* indicates that the method is both robust and efficient, even for low-quality triangulations.

CCS Concepts: • **Computing methodologies → Shape modeling**.

Additional Key Words and Phrases: geodesic, edge flip, triangulation

这是一篇 SIGGRAPH 2020 上的文章，属于 Intrinsic Triangulations 的一系列文章之一。当几何物体表面的三角形比较均匀时，我们在上面进行一些几何处理（比如表面参数化、拉普拉斯光滑等）往往能得到比较好的效果。但是现实中的三角形网格可能有大量的 ill-condition 的三角形：



比如上图中有非常多细长的三角形，这会导致几何处理的结果出现不自然的结果。Intrinsic Triangulations 解决的就是如何在这样的三角形网格上做各种几何处理的问题。作者在 SIGGRAPH 上关于 Intrinsic Triangulations 做了一次非常详细的入门教程，同时配有开源代码：

https://www.youtube.com/watch?v=gcRDdYrgOhg

你可以首先学习这个教程，阅读 paper 理解算法的原理，然后尝试复现论文中的算法。

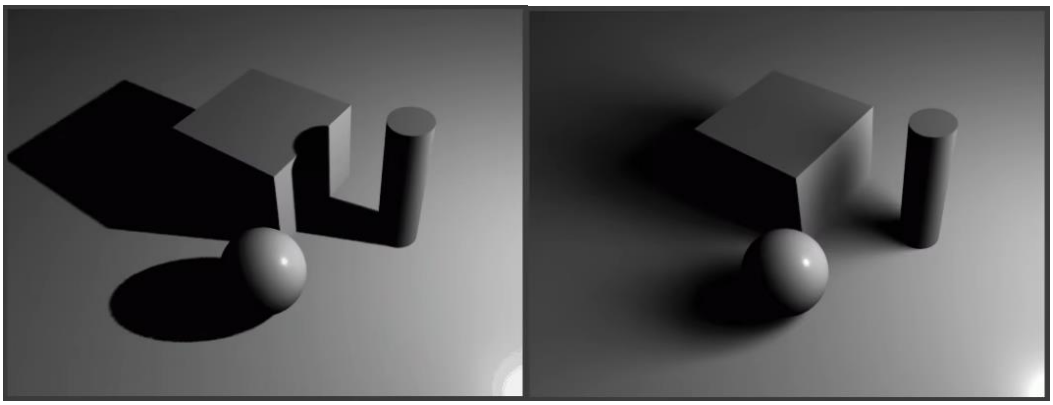
C. Rendering

1. Better Real-Time Shadow (★)

https://sites.cs.ucsb.edu/~lingqi/teaching/games202.html

https://www.bilibili.com/video/BV1YK4y1T7yY

Lab3 中我们实现了用光栅化方法渲染阴影，但是阴影的边界比较生硬。现代游戏往往需要更真实的光影效果，比如实时软阴影：



注意上图中右边柔和的阴影边界以及强度的渐变。在这方面有非常多的工作可以参考，b 站上闫令琪老师开设的《GAMES202-高质量实时渲染》是很好的入门教程。

2. Better Non-photorealistic Rendering (★)

Lab 中实现了一种简单的非真实感渲染的方法，为了实现不同的艺术效果有非常多的非真实感渲染的技术，比如游戏《原神》中的 “三渲二” 技术：

这些艺术风格的实现往往需要综合使用 geometry processing、shading、post processing 的各种方法。请自由探索自己想实现的艺术风格，然后研究原理并在 lab 的代码基础上实现出来。

3. SDF Text Rendering (★)

https://steamcdn-a.akamaihd.net/apps/valve/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf

Improved Alpha-Tested Magnification for Vector Textures and Special Effects

Chris Green*
Valve

(a) 64x64 texture, alpha-blended (b) 64x64 texture, alpha tested (c) 64x64 texture using our technique

这是一篇 SIGGRAPH 2007 的文章，作者提出一种使用 SDF 在光栅化管线中渲染文字的方法。请研究算法的原理并在 lab 的代码基础上复现出来。

4. Ray Tracing Acceleration (★ ★)

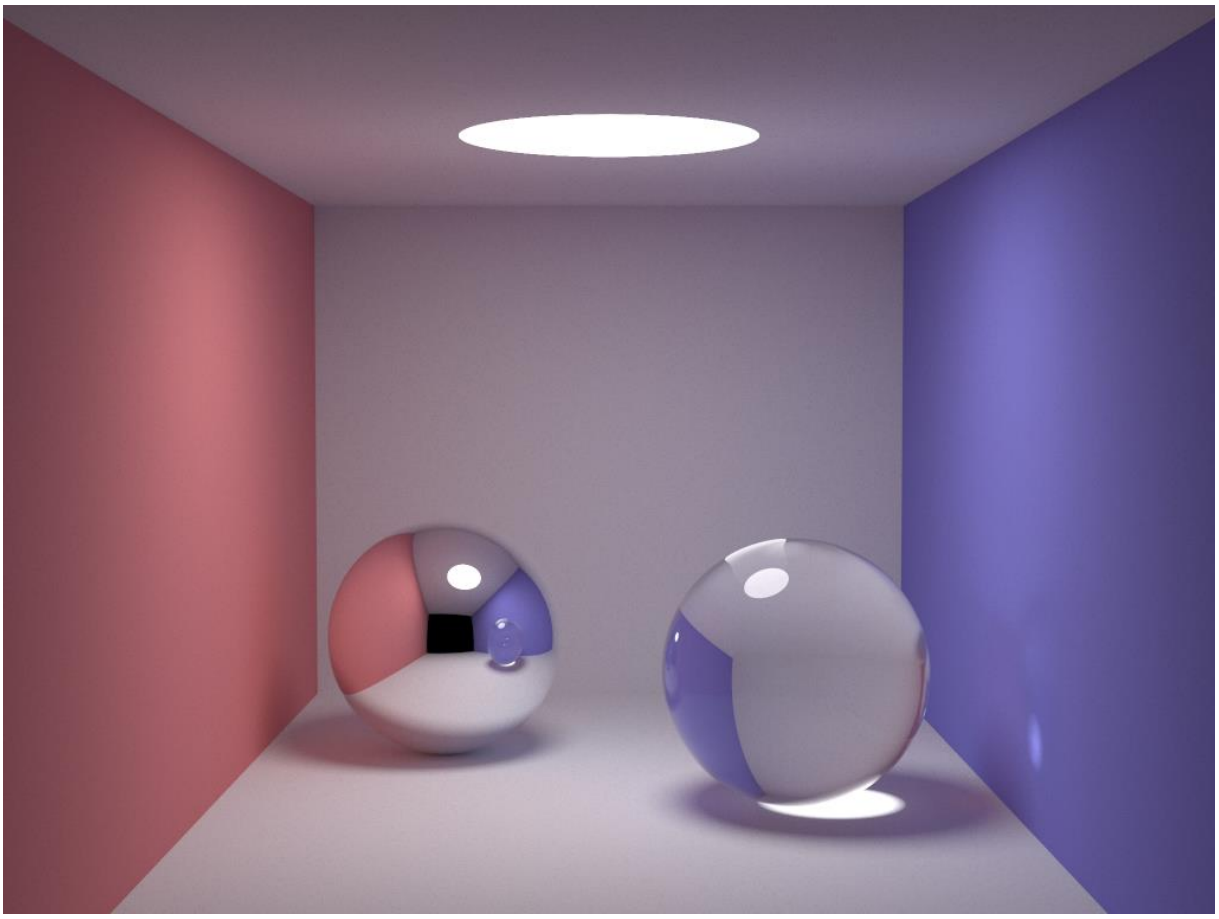
在 lab3 中，大家可以发现光线追踪的渲染效果要好于光栅化，但是花费的渲染时间要远多于光栅化。利用加速结构，比如 AABB Tree、BVH Tree，Spatial Hashing 等，加速光线追踪采样是常见的做法。请研究并实现光线追踪的各种加速结构，乃至 CPU、GPU 并行，实现更快的光线追踪。

5. Path Tracing (★ ★)

http://www.kevinbeason.com/smallpt/

https://www.bilibili.com/video/BV1X7411F744

Path tracing 不同于 lab3 中实现的 Whitted-style ray tracing, 是基于渲染方程的全局光照渲染方法，是现代基于物理真实感渲染的基础。Smallpt 是 99 行的 C++程序，但是能渲染出下面的效果：



注意图中的阴影效果是非常柔和的，而 lab3 中的阴影边界是非常锐利的。请研究 Path tracing

的原理并在 lab 的代码基础上实现。b 站上闫令琪老师开设的《GAMES101-现代计算机图形学入门》是很好的入门教程。

6. Photon Mapping (★ ★)

Photon mapping 是除 path tracing 之外的另一种全局光照算法，现代专业渲染器往往会把 photon mapping 和 path tracing 作为一个选项提供给大家，同样能渲染出非常漂亮的光影效果：

你可以首先调研 photon mapping 的原理，然后尝试在 lab 代码的基础上复现出来。

7. Advanced Rendering Techniques (★ ★ ★)

如果你对渲染的理解已经非常深入了，欢迎尝试实现各种高级的渲染技术来惊艳助教们。这些技术包括光栅化中的 Deferred Rendering、Light Probs、Atmosphere Rendering 等等，离线渲染中的 Bidirectional Path Tracing、Metropolis Sampling、Volume Rendering 等等，以及利用现代图形 API 实现 Real-Time Ray Tracing 等等。

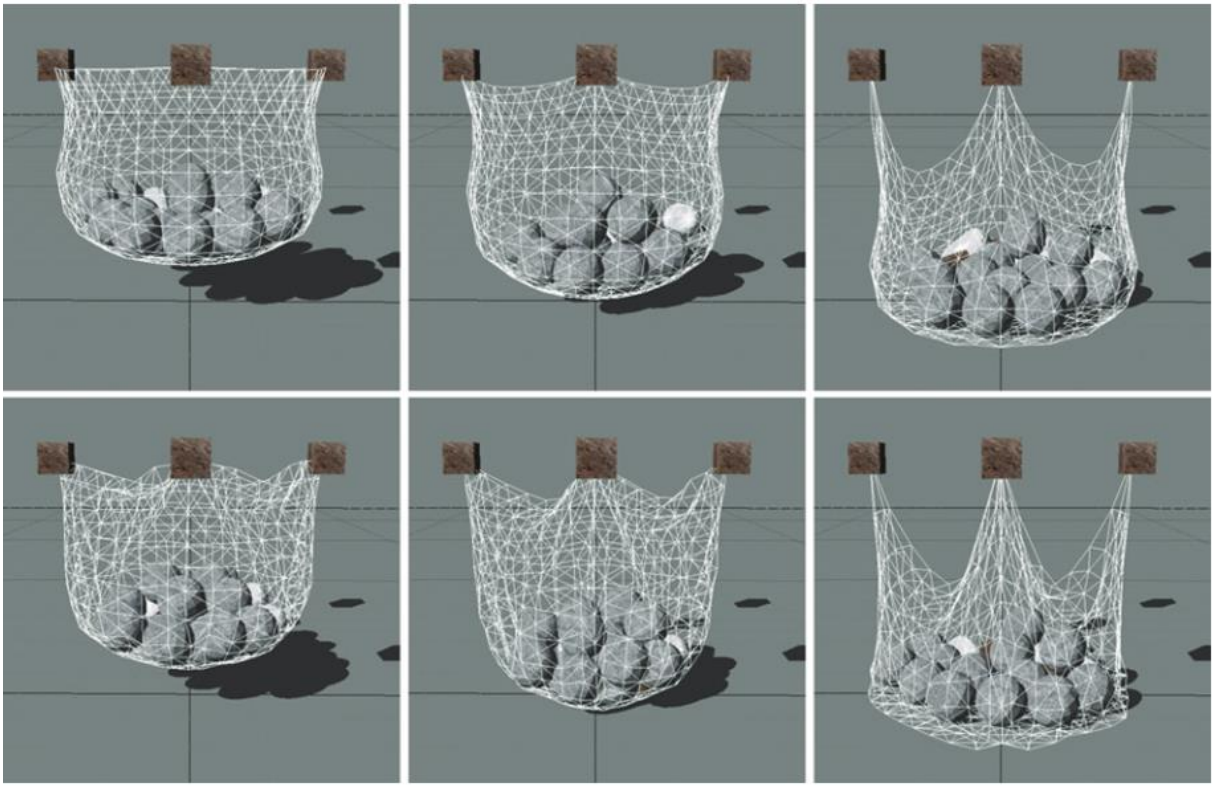
D. Simulation

1. Position Based Dynamics (★)

https://www.bilibili.com/video/BV12Q4y1S73g

A Survey on Position-Based Simulation Methods in Computer Graphics

https://onlinelibrary.wiley.com/doi/epdf/10.1111/cgf.12346

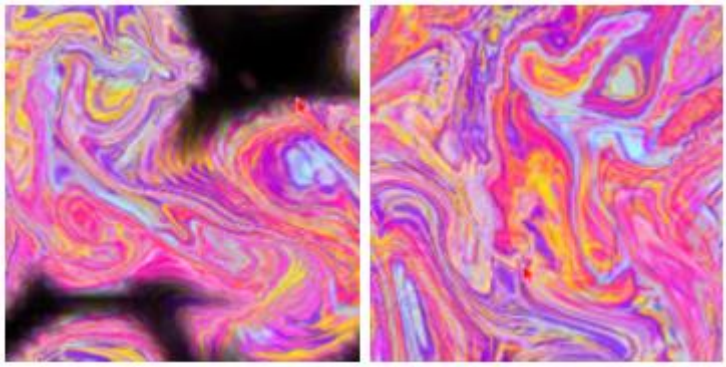


Position Based Dynamics(PBD) 因其简单性和稳定性，是游戏中最常用的实时物理模拟方法，可以模拟各种材料以及之间的交互，包含弹性体、刚体、流体等等。你可以从 PBD survey 或者网络上的某一个教程入手，学习 PBD 算法的原理，然后在 lab 代码的基础上复现出来。

2. Fluid Simulation (★★)

如何在计算机中模拟流体是一个经典的问题。我们提供两种最经典的算法给大家参考：一种是欧拉网格模拟方法，经典文章是 SIGGRAPH 1999 上提出的 Stable Fluids 方法

https://www.dgp.toronto.edu/public_user/stam/reality/Research/pdf/ns.pdf



另一种是光滑粒子流体动力学法（SPH），可以参考下面的课程入门：

https://interactivecomputergraphics.github.io/SPH-Tutorial/

3. Fast Simulation of Mass-Spring Systems (★ ★ ★)

http://tiantianliu.cn/papers/liu13fast/liu13fast.html

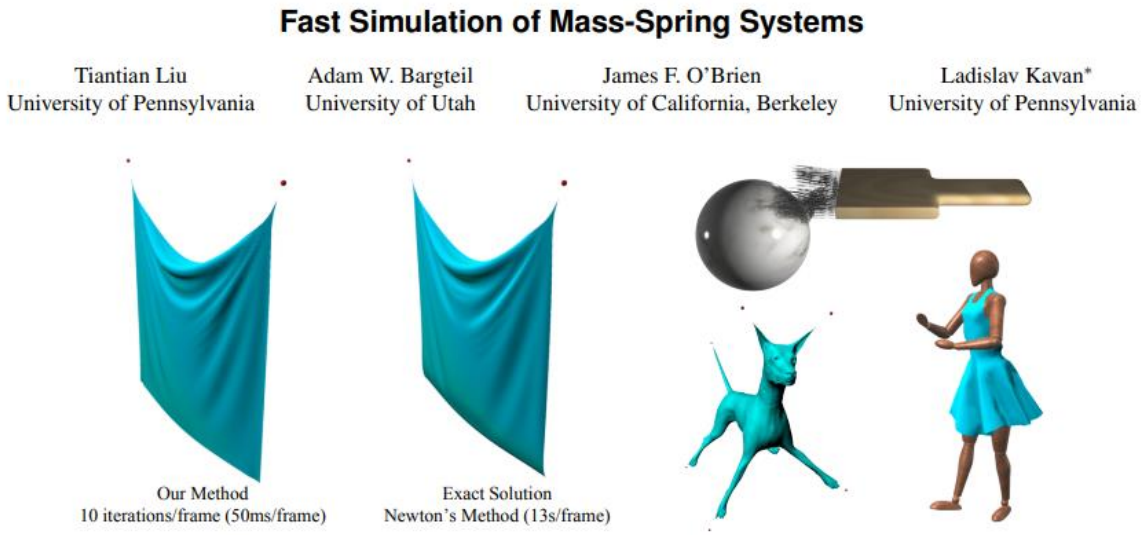


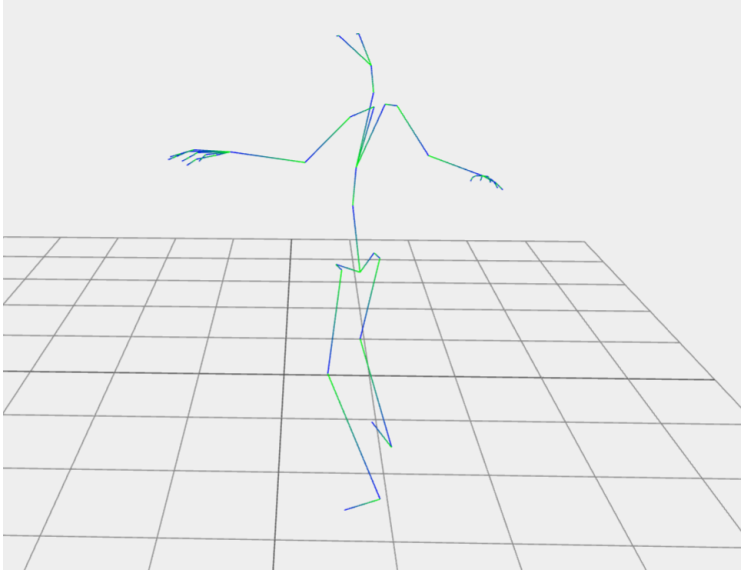
Figure 1: *When used to simulate the motion of a cloth sheet with 6561 vertices our method (left) produces real-time results on a single CPU comparable to those obtained with a much slower off-line method (middle). The method also performs well for one dimensional strands, volumetric objects, and character clothing (right).*

lab4 中我们实现了隐式弹簧质点的求解，其中每一帧都求解一个线性系统的开销是非常大的。在 SIGGRAPH 2013 上，作者提供了一种简单并有效的加速弹簧质点解算的方法：通过 local-global 的交替迭代并预分解矩阵，能够大大加速弹簧质点系统的解算。

E. Animation

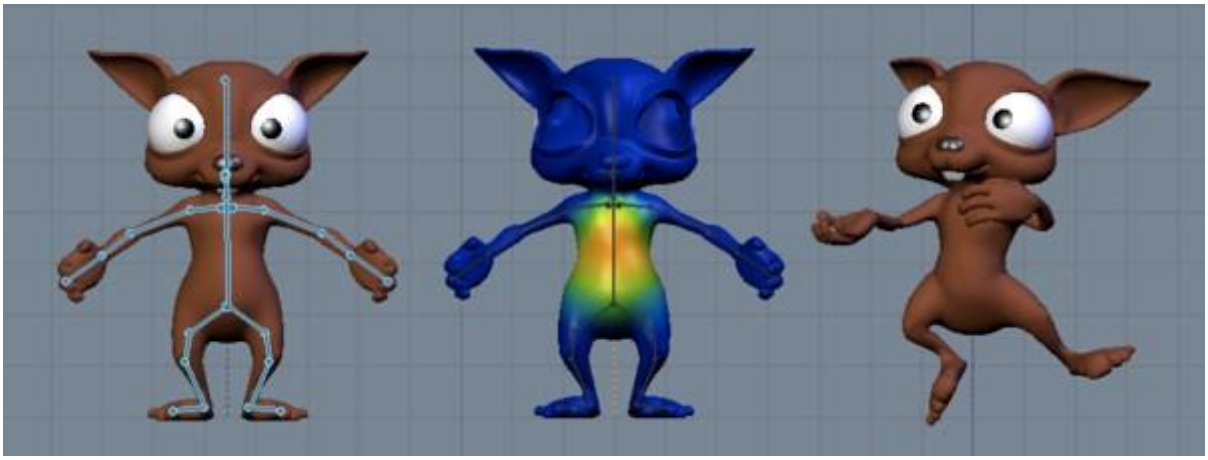
1. BVH Loader (★)

https://threejs.org/examples/webgl_loader_bvh.html



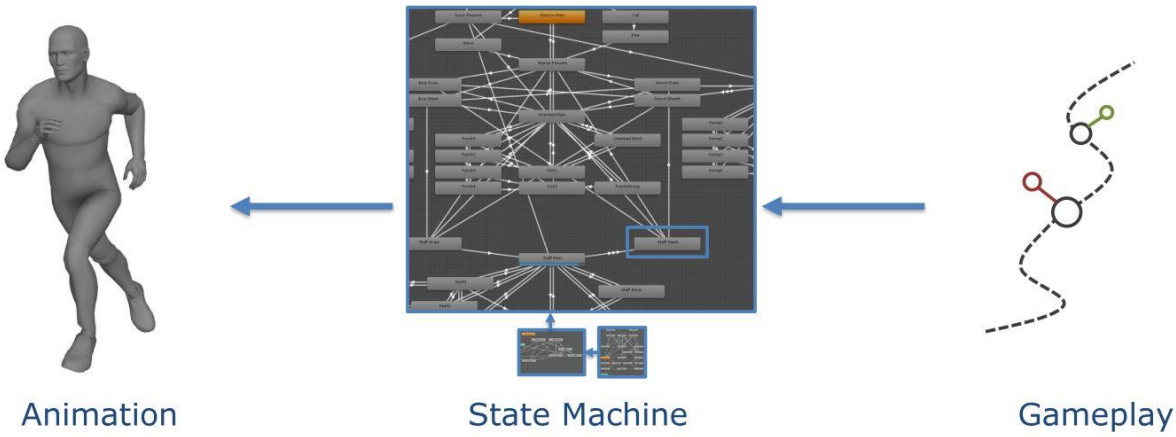
BVH 文件是最常用的描述人体动作的文件格式，里面记录了各个关节以及关节的渲染。请在 lab 代码的基础上实现一个 BVH 文件的加载器，实时显示出人体的动作序列。

2. Skinning (★★)



像 BVH 这样的骨骼动画文件只描述了骨骼的运动，而角色的蒙皮需要骨骼驱动一起运动，这个过程就是 skinning。skinning 是经典的动画领域问题，你可以先学习经典的蒙皮算法的原理，然后在 lab 的代码基础上实现实时的蒙皮效果。当然，在这之前你需要有一个 BVH Loader。

3. Motion Matching (★ ★ ★)

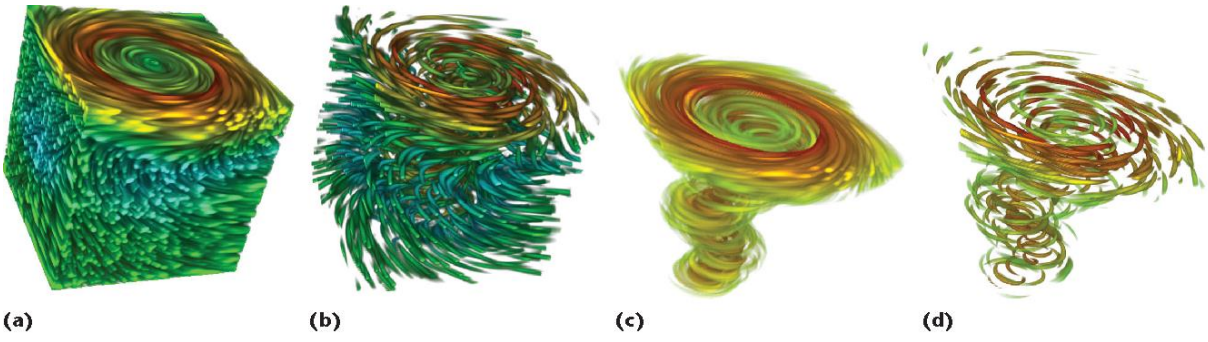


在游戏中，我们通过键盘和鼠标控制角色跑、转弯、走等不同动作，这些不同动作之间的自然过渡需要算法实现，motion matching 就是其中一个经典的算法。你需要学习 motion matching 的原理然后在 lab 代码基础上复现出来。当然，在这之前你大概率需要一个带蒙皮的 BVH loader。

F. Visualization

1. Better Flow Visualization (★)

http://www.zhanpingliu.org/Research/FlowVis/FlowVis.htm



Lab 中我们实现了基于 LIC 的 2D 流场可视化方法。LIC 方法有非常多的改进工作，比如如何让

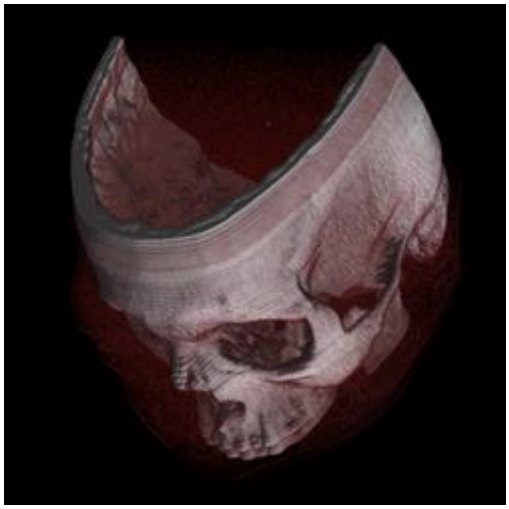
可视化的结果能指示方向，如何动态可视化，如何可视化三维流场。请探索不同的可视化流场方法，并分析各自的优劣。

2. Word Cloud (★ ★)

Word Cloud 是经典的可视化方法，如上图所示：出现频率高的单词会更大，单词们可以组合成一个有意义的图形。请研究生成 Word Cloud 背后的算法，然后复现出来。

3. Volume Visualization (★ ★ ★)

https://en.wikipedia.org/wiki/Volume_rendering



可视化体积信息有着广泛的应用，比如用于医疗中可视化骨骼、肌肉等组织，其背后使用的是 Volume Rendering 的技术。请探索这个技术背后的原理，并复现出体积信息可视化的结果。