Kasumi Renderer 核心架构

Platform（图形渲染后端平台）

- App（抽象类，注册进Platform）
  - 生命周期
    - virtual void prepare()
    - virtual void update(double dt)
    - virtual auto quit() -> bool
  - 回调函数
    - virtual void key(int key, int scancode, int action, int mods)
    - virtual void mouse_button(int button, int action, int mods)
    - virtual void mouse_scroll(double x_offset, double y_offset)
    - virtual void mouse_cursor(double x_pos, double y_pos)
- 回调函数集合
- Renderer App（实例一：Kasumi渲染器）
  - 实现Manager回调，并向上传递
  - Manager：UI
  - 实现场景输入回调，并向上传递
  - Scene：场景
  - 相机回调实现，并向上传递
    - 场景相机：Camera
    - 场景加载/保存
      - auto read_scene(const std::string &path) -> std::string
      - auto write_to_file(const std::string &path) -> std::string
    - 场景数据管理
      - inline auto get_camera() -> CameraPtr &
      - inline auto get_object(unsigned int id) -> SceneObjectPtr
      - auto add_object(ModelPtr &o) -> unsigned int
      - auto add_object(ModelPtr &&o) -> unsigned int
      - void erase_object(unsigned int id)
      - void restore_object(unsigned int id)
    - 场景物体数据集合：<unsigned int, SceneObjectPtr>
  - Api：抽象组件，支持外部注册（如物理模块）
    - 生命周期
      - virtual void prepare()
      - virtual void step(float dt)
    - UI
      - virtual void ui_menu()
      - virtual void ui_sidebar()
      - virtual void key(int key, int scancode, int action, int mods)
    - ScenePtr：内含场景指针，支持访问场景数据

Model（几何模型数据）

- 模型加载方法
  - 模型绝对路径：Model(const std::string &model_path)
  - 内置模型加载（附带纹理）：Model(const std::string &primitive_name, const std::string &texture_name)
  - 内置模型加载（指定颜色）：Model(const std::string &primitive_name, const mVector3 &color)
  - 自定义几何数据加载（可附带纹理）：Model(std::vector<UniversalMesh::Vertex> &&vertices, std::vector<Index> &&indices, std::map<std::string, std::vector<TexturePtr>> &&textures = {})
- UniversalMesh：通用Mesh类，向外封闭，不允许在外部调用
  - 顶点：std::vector<Vertex>
  - 三角面：std::vector<Index>
  - 纹理：std::map<std::string, std::vector<TexturePtr>>
- Shader
  - 默认MeshShader
  - 默认InstancedMeshShader
  - 默认LineShader
  - 支持使用自定义Shader：void use_custom_shader(const ShaderPtr &shader)
- 功能：实例化渲染
  - 启用实例化：void instancing()
  - 添加多个实例化物体：void add_instances(const std::vector<Pose>& poses)
  - 添加一个实例化物体：void add_instances(const Pose& pose)
  - 清除所有实例化物体：void clear_instances()

作为整个几何数据集合向外解耦，封闭在SceneObject内。

作为**可渲染对象**，存储进SceneObject