开发笔记

Ivan Lin

2017 年 2 月 3 日

Visual Studio

Resharper 插件
alt + o: .h 和.cpp 文件切换
alt + 鼠标: 框选模式
ctrl+k + ctrl+c: 注释代码
shift+alt+up/down: 框选模式上下
ctrl+alt+a: open Command Window
ReSharper_Suspend/ReSharper_Resume in Command Window: close/open ReSharper
lib: 静态库，在编译时将库代码加入程序中；dll: 动态库，编译时生成一个 lib 和一个 dll，lib 用于存放 dll 中相应接口的索引

计算机图形学

坐标系模拟：拇指 x，食指 y，中指 z。左手系和右手系
标准化向量 = 单位向量 = 法线，$\mathbf{v}_{norm} = \frac{\mathbf{v}}{|\mathbf{v}|}$
a + **b** 几何解释：**a** 的头连接 **b** 的尾，然后从 **a** 的尾向 **b** 的头画一个向量
a - **b** 几何解释：**a** 的尾连接 **b** 的尾，然后从 **b** 的头向 **a** 的头画一个向量
向量点乘：$\mathbf{a} \cdot \mathbf{b}(\mathbf{ab}) = a_1b_1 + ... + a_nb_n$, 几何解释：$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$（两向量夹角）
向量投影：**v** 分解为平行和垂直于 **n** 的两个分量。

$$\mathbf{v}_{||} = \mathbf{n}\frac{\mathbf{v} \cdot \mathbf{n}}{|\mathbf{n}|^2} \qquad \mathbf{v}_{\perp} = |\mathbf{v}| - \mathbf{v}_{||}$$

向量叉乘：仅可用于 3D 向量，$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} \mathbf{a}_y\mathbf{b}_z - \mathbf{a}_z\mathbf{b}_y \\ \mathbf{a}_z\mathbf{b}_x - \mathbf{a}_x\mathbf{b}_z \\ \mathbf{a}_x\mathbf{b}_y - \mathbf{a}_y\mathbf{b}_x \end{bmatrix}$，几何解释：结果向量垂直于原来两个向量，$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}||\mathbf{b}|sin\theta$
, $|\mathbf{a} \times \mathbf{b}| = 0$ 表示 $\mathbf{a}$ 与 $\mathbf{b}$ 平行或有一个为 $\mathbf{0}$
矩阵转置：M^T, 其列由 $\mathbf{M}$ 的行组成，$M^T{}_{ji} = M_{ij}$
$(AB)^T = B^T A^T$, 可推广到字符串翻转
$P_{camera} = P_{object} M_{object \to world} M_{world \to camera}$
线性变换：F(a+b) = F(a)+F(b), F(ka) = kF(a), 则称映射 F 是线性的（$\mathbf{aM}$ 满足此条件）
仿射变换：线性变换后接平移, $v^{'} = v\mathbf{M} + \mathbf{b}$
对 $\mathbf{aM}$, 求逆变换等价于求矩阵的逆
矩阵行列式：$|\mathbf{M}| = \sum_{j=1}^{n} m_{ij} c_{ij} = \sum_{j=1}^{n} m_{ij}(-1)^{i+j} |\mathbf{M}^{\{ij\}}|$
矩阵的逆: $M(M^{-1}) = M^{-}1M = I$, 不可逆矩阵又称奇异矩阵，奇异矩阵行列式为 0
标准伴随矩阵：adj$\mathbf{M}$, M 的代数余子式矩阵的转置矩阵。$M^{-1} = \frac{adj\mathbf{M}}{|\mathbf{M}|}$
正交矩阵：$MM^T = I$, 旋转和镜像矩阵是正交矩阵。正交矩阵满足：矩阵的每一行都是单位向量，矩阵的所有行相互垂直。
Vector4, 齐次坐标。(x, y, z, w) 实际代表 3D 中的 (x/w, y/w, z/w)
旋转矩阵：描述一个坐标中基向量到另一个坐标基向量的转换。
矩阵蠕变：由于浮点数精度有限导致误差积累。
欧拉角：heading-pitch-bank 约定。
万向锁：三个角度不互相独立，一旦选择正负 90 度为 pitch 角，就被限制在只能绕竖直轴旋转，失去了一个维度。
四元数：
几何图元自由度：是无歧义的描述该实体所需信息量的最小数目。
射线：是一个有向线段，参数形式：$x(t) = x_0 + t\triangle x \quad y(t) = y_0 + t\triangle y$, 向量记法:$p(t) = p_0 + td$, 增量向量 d 指定了它的长度和方向, 斜截式：$y = mx + b$
球和圆：

Sublime Text 2

ctrl+shift+up/down: move line up/down
ctrl+alt+up/down: block edit up/down

Swift

http://blackblake.synology.me/wordpress/?p=29: Swift 里的 Optional 和 Unwrapping

PhotoShop

alt+ctrl+c: Resize Canvas
alt+ctrl+shift+s: Save for web
ctrl+h: show canvas guides; ctrl+mouse drag ruler: add canvas guide

LaTeX

%!Mode:: "TeX:UTF-8": make WinEdt show Chinese

Git

gitk file/folder: show commit with file

Windows

放大镜:ctrl+alt+d: 停靠模式; ctrl+alt+l: 窗口模式; win++: 放大; win+esc: 退出放大镜

JavaScript

JavaScript 组成: ECMAScript, DOM: 针对 XML 文件的操作接口, BOM: 浏览器对象模型，HTML5 标准化
浮点数误差：0.1 + 0.2 = 0.300000000004，通过 x10 法解决
JS 没有函数签名，所以不能重载，可通过判断输入参数 (arguments, callee, caller) 的个数模仿重载. P.S. 函数签名: 包含函数名、参数类型、类名以及空间名等
JS 对象类型：基本类型和引用类型，不可以直接操作内存空间，操作的是对象的引用。
JS 函数参数是值传递。
JS 没有块级作用域，即 if/for 循环中的变量不会在循环结束后销毁。局部环境：function, 全局环境
垃圾回收机制：1. 标记清除；2. 引用计数（会有循环引用的问题）
引用类型不是类，类是定义同一类所有对象的变量和方法的蓝图或原型，有接口和结构。
推荐使用对象字面量语法定义对象，因为更有封装概念，并且可以“重载”构

造函数。
JS 的数组可以同时储存任何类型的数据，且长度动态增长。var colors = new Array(); || var colors = [”red”, ”green”, ”blue”];
JS 的数组 length 是非只读的，修改该值会删除相应的元素。
splice 函数: splice(0, 2): 删除数组中的前两项; splice(2, 0, ”red”, ”green”): 从位置 2 插入”red” 和”green”; splice(2, 1, ”red”): 删除位置 2 的元素，并插入”red”
JS 的函数实际上是 Function 类型的实例。
解释器会先读取函数声明 (function sum();)，但函数表达式 (var sum = function();) 需要到相应的行才会执行。
this 引用的是函数据以执行的环境对象。
不属于其他任何对象的属性和方法都属于 Global 对象的属性和方法，如 isNaN().
eval() 函数传入完整的 JS 代码。
代码注入：
对象中数据属性的四个特性:[[Configurable]] [[Enumerable]] [[Writable]] [[Value]], 使用 defineProperty(obj, ”propertyName”) 函数修改。