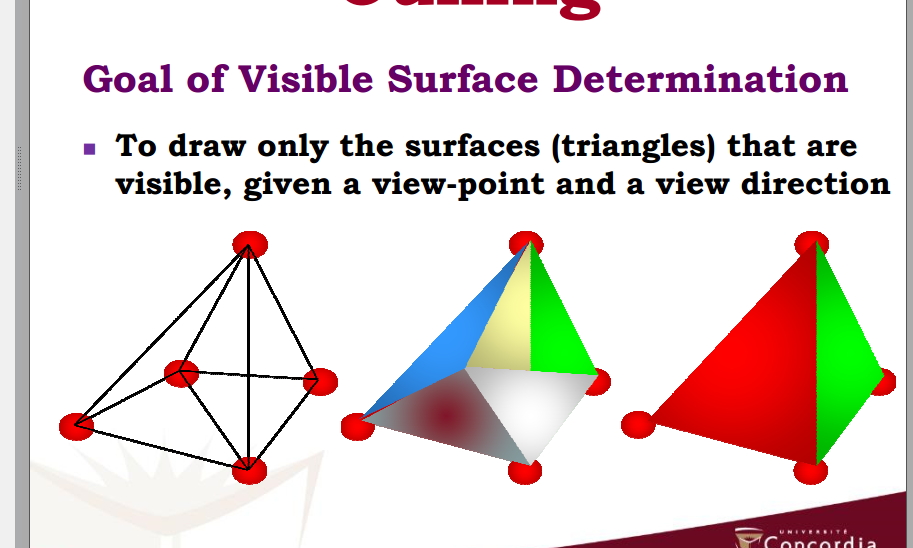
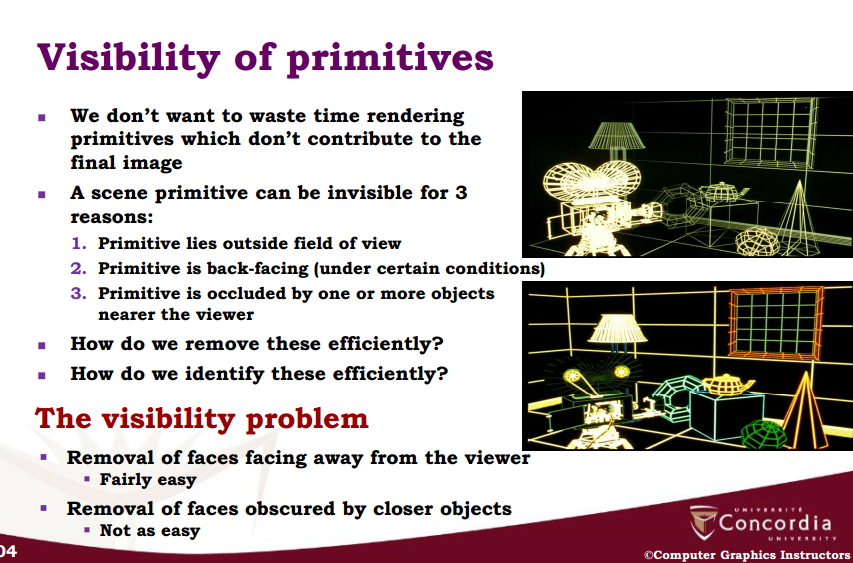
Culling 选择，剔除劣质货



中间的那个是所有面都画出来，糊成一团

右边的那个是culling过的，只画可以看见的三角形。 者必须建立在有一个观察点以及观察方向的基础上‘’



图形的可见性

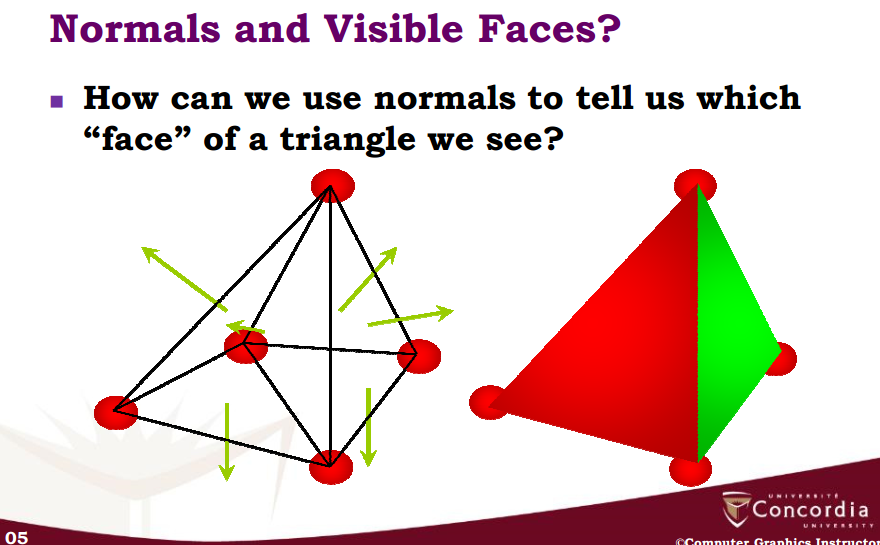
，我们不想画那些看不出来的Primitive

三个原因决定他看不看得见

1.在视角边界以外的primitive

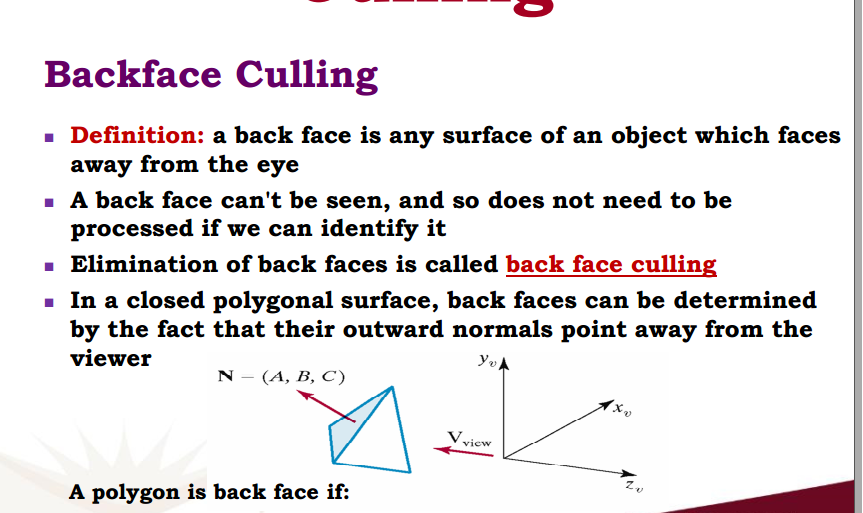
2.背朝你的primitive

3. 原来的primitive被一个或多个object遮挡



我们怎样使用Normal知道看见的是正面还是背面

‘答案，使用back culling



Back face指背对camera的面

Back face不能看见，所以我们没必要处理它

消除back face的过程叫做back face culling

一个封闭的几何图形，是不是back facce可以通过他们的normal是不是指向viewer

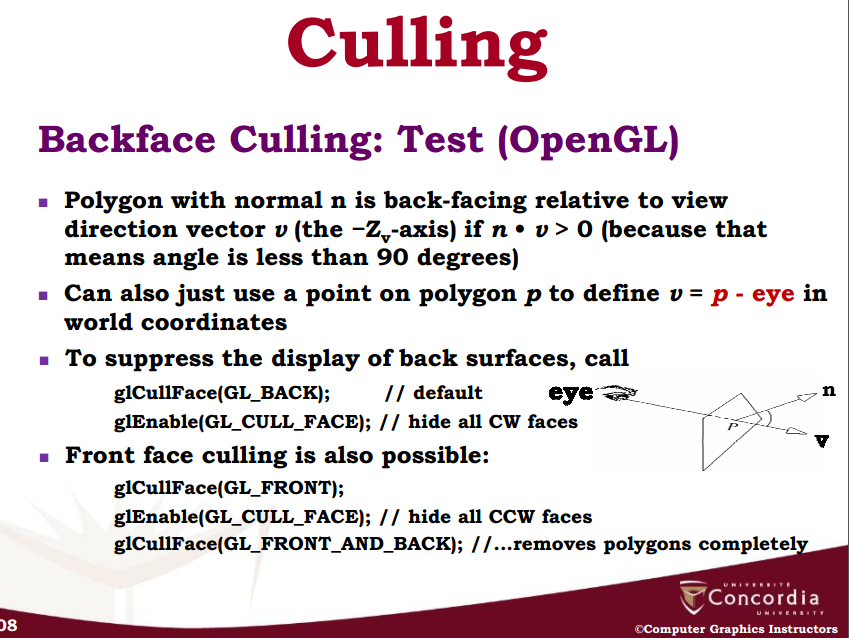


只要vector view \* N >0,

vector view代表着视角的向量，

N代表着这个面的向量，

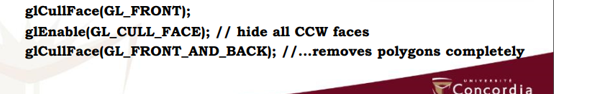
乘积大于零代表着两个向量方向是相同的代表着背对

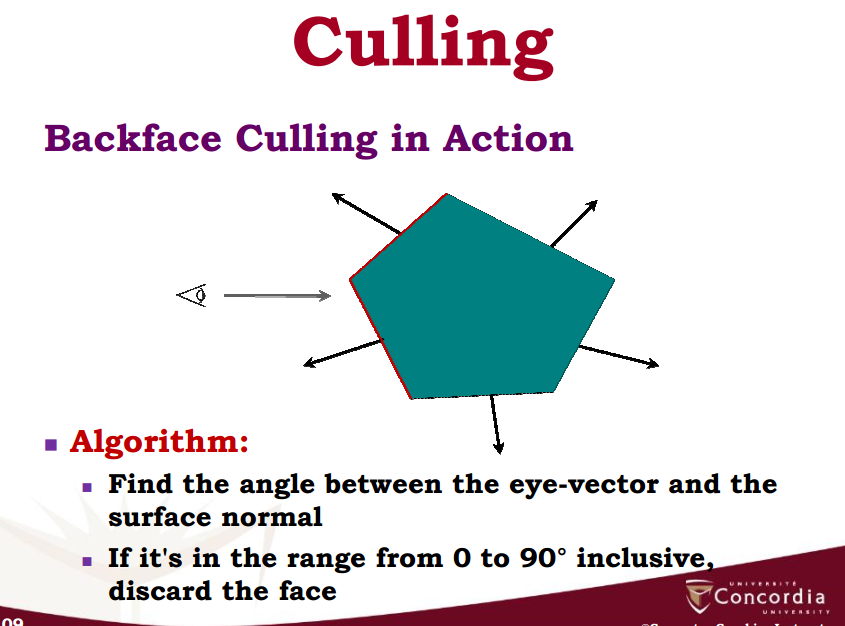


一个polygon是否back-facing取决于normal与view 方向是否一致，大于0代表着夹角小于90度

怎么样抑制suppress展现back surface

怎么甚至把朝着viewer的face也屏蔽//导致polygon彻底无法显示



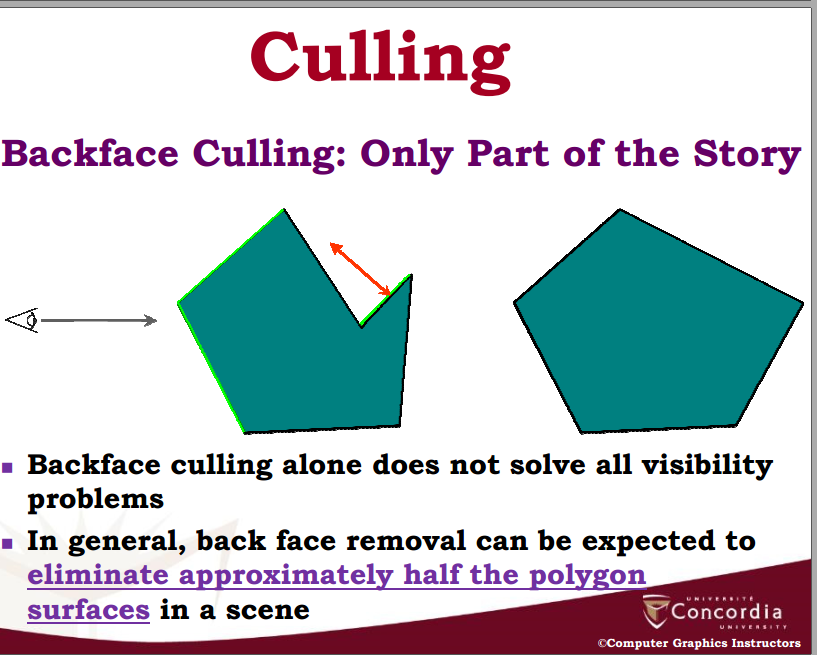


实际算法就是

找到eye-vector与surface normal之间的角度

如果角度小于90°

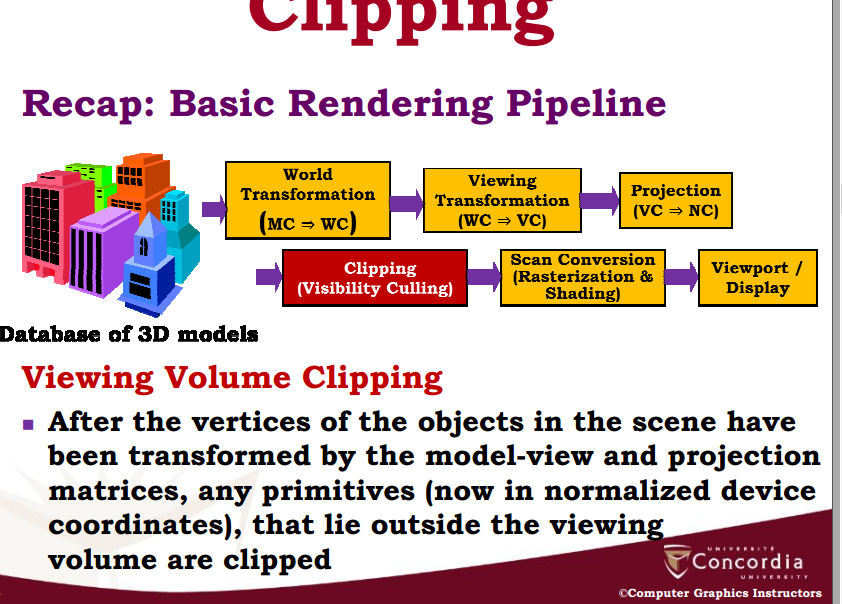
那么就忽略这一面



但backface culling只是一部分的拼图

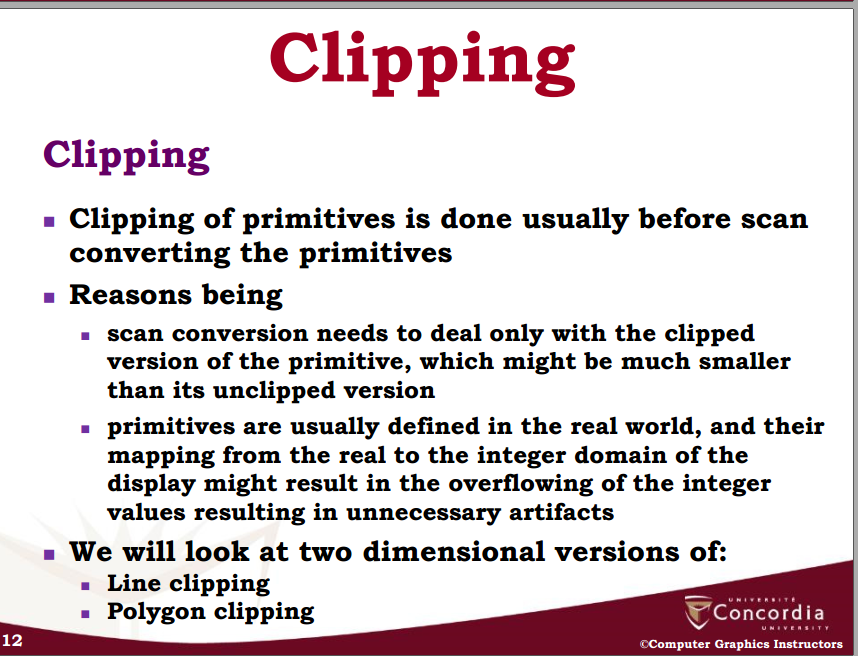
通常来说，根据大量经验，back face removal大概能移走一半的polygon surface

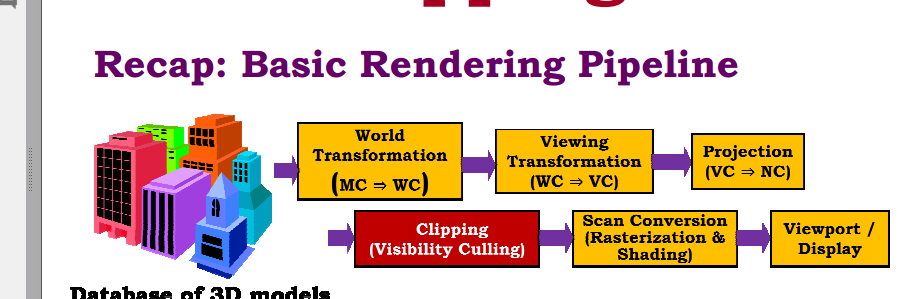
Clipping 剪裁



当所有object都被转换成model-view与projection mattices

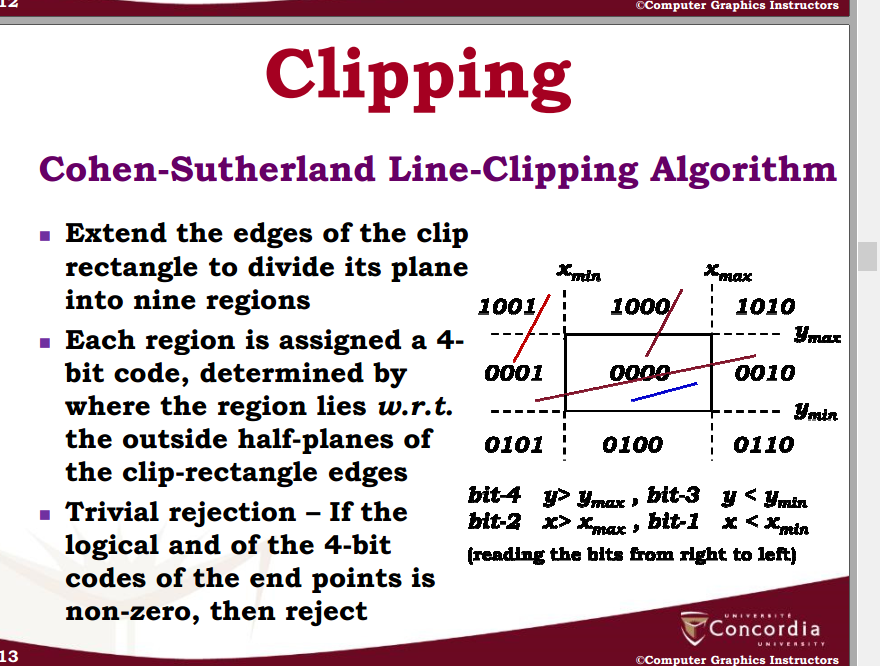
任意在viewing volume之外的primitives都会被clip





对primitive 的clipping通常在vertice已经定位好之后，但是还没有上色

这样就可以对那些view volume之外的 vertice不上色，节省资源



Line-clipping:cohen sutherland

把切割的长方形 extend edge，切割出九部分

每一个部分给他一个四位二进制代码（生成TBRL code, top bot right left）

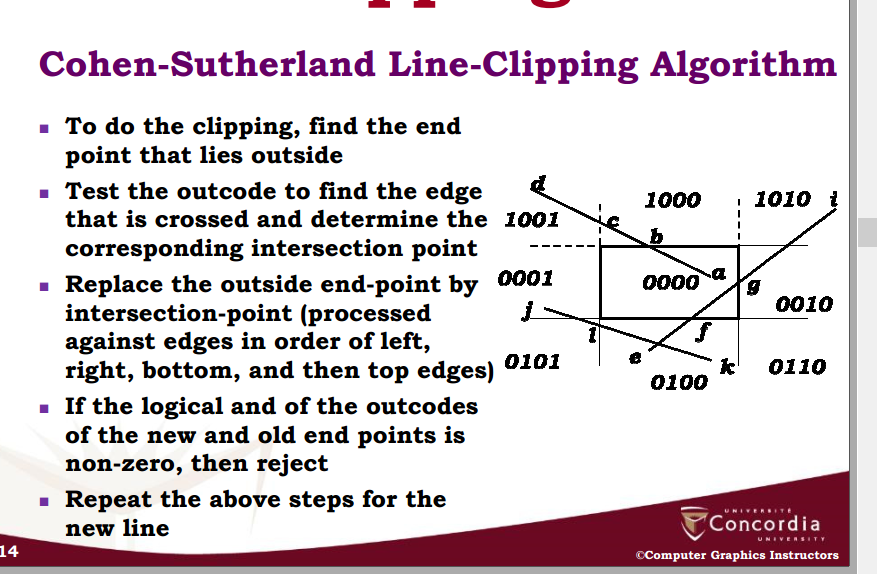
这是二维空间

从右往左数第四位有东西（1）代表着Y超过max

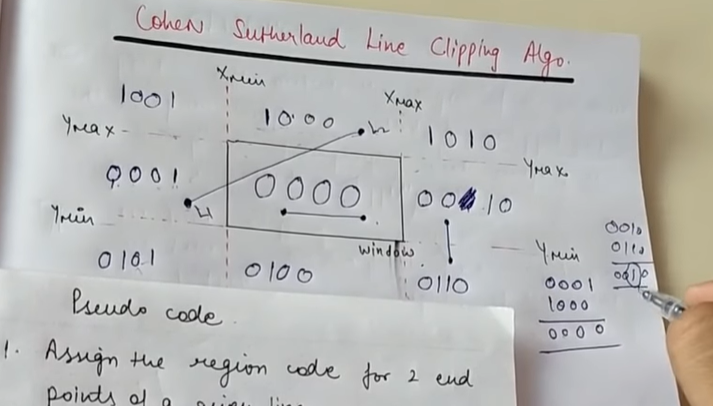
第三位代表着小于Ymin

第二位代表着X大于XMAX

第一位代表着小于Xmin



怎么clipping

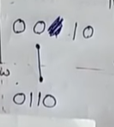


首先两个end point都在0000之间的线段，完全容纳

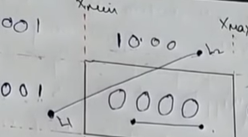
然后找到在外面的end point，

然后对着两个end point进行 AND operation(只有两个都是1才是1)

如果两个end point结果最后不是0000，舍去，说明完全没经过我们核心地带0000



结果是0000，说明有可能inside

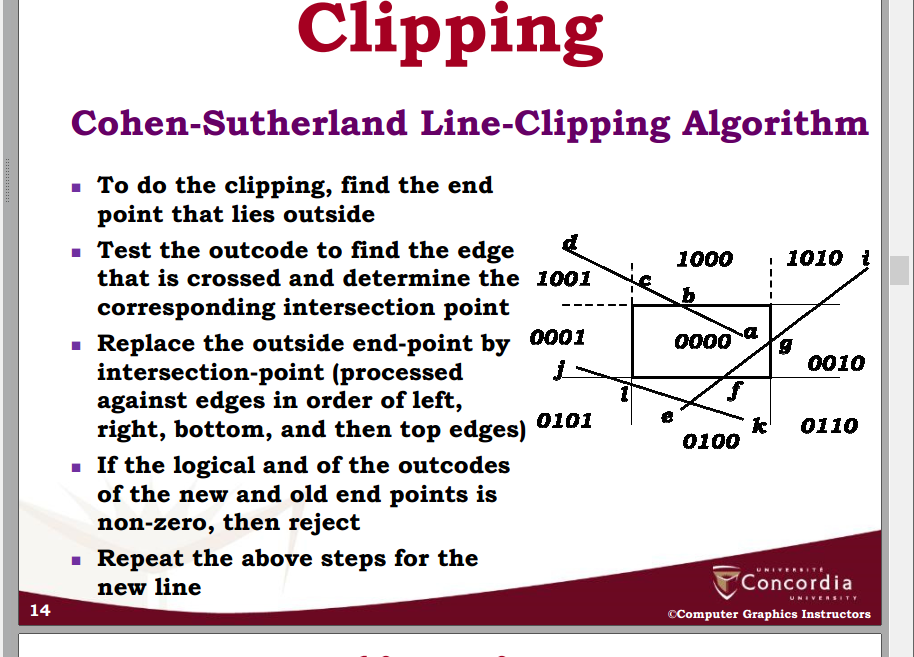


因为你可以把这个线段往上一点点，有可能结果是0，没有交叉

然后决定intersection point，也就是LRBT四条直线（没距离）找焦点，用新点代替旧点

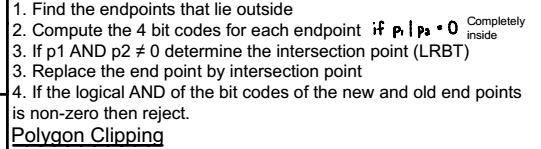
例如这里L1原来是0001，与LEFT交叉过以后就是0000，（注意1只是大于XMINXMAX的情况，在这条直线上就是0）

然后对这些交叉过的新点使用AND逻辑，如果还不是0，就舍去，



这里就是例子：

首先找到outside的点,例如d，找到edge da，先经过LEFT线，形成AC，test and逻辑，还是可以



首先排除那些彻底在外面的线段（通过AND不是0）

然后进行这个循环

如果两个端点都是0，completely inside

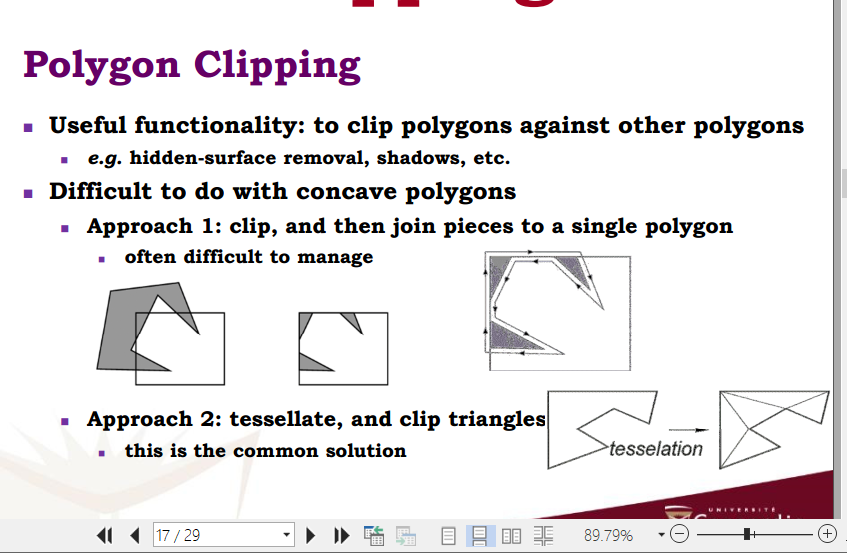
然后如果任意一个不是0，，第一步交叉，与TBRL碰撞，截取一部分

让新的点代替旧的点，然后让新的点与旧的点进行logical，如果答案不是0，就可以舍去了，例如cd

Polygon clipping

多边形裁剪

首先我们可以用hidden00surface removal等技巧来让polygon之间互相切割，然后再将结果送往windows



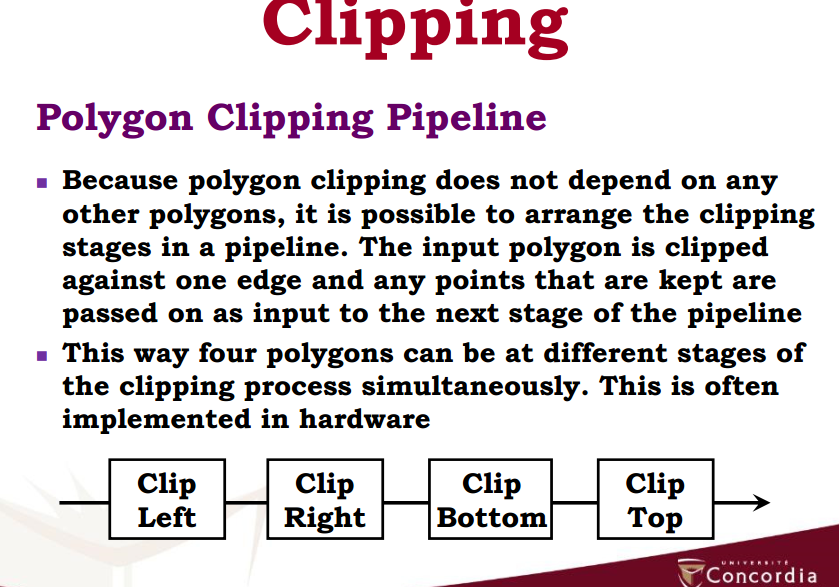
把视野外的多边形，对于concave凹型躲避线性有些复杂

第一种解决方案，裁剪，然后把里面的裁剪部分（因为是凹的所以会断开），然后把这三个连接起来形成一个新的polygon(难以管理)

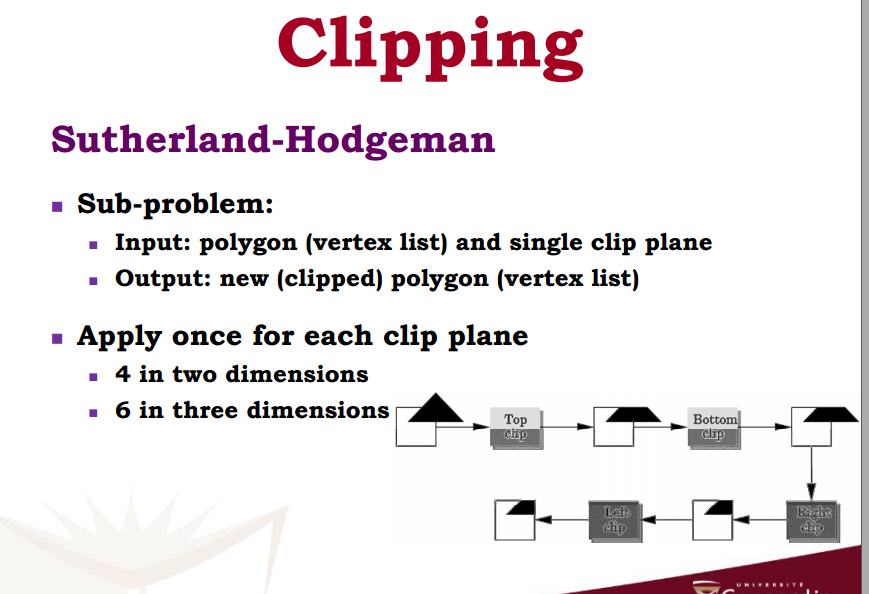
第二种解决方案：常规解决方案，把这个多边形tessellate 剪切 成多个三角形

三角形就不是凹边形，就可以比较容易的clipping

然后我们将简单图形与长方形（window）来clip



切割方案是通过edge来的，所以上下左右实际上是四个不同阶段，只有经过左右上下这个pipeline才算完成切割



Sutherland-hodgeman

将polygon于长方形polygon相clip的技巧（使用cohen-sutherland line cliipping法来一条一条切割edge）

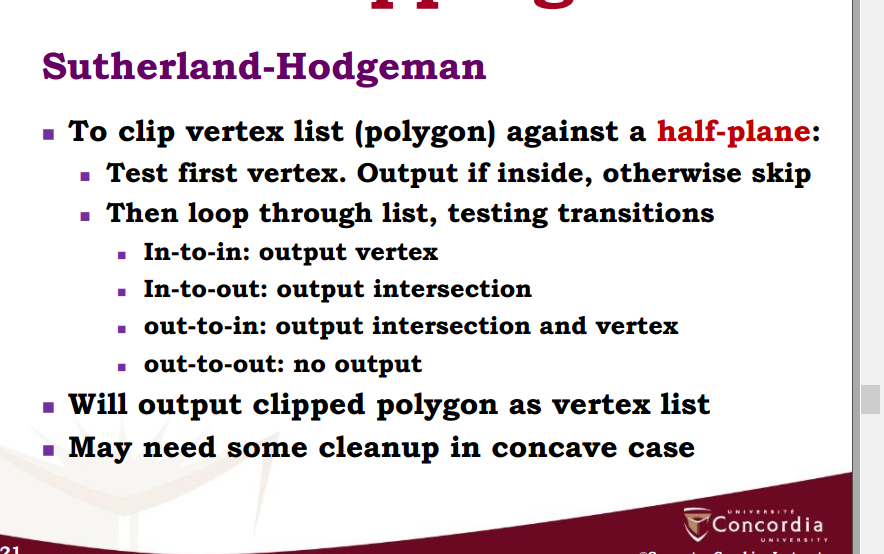
输入一个polygon(一个vertex array) 与一个切割面（view volume）

输出一个新切割好的polygon(vertex array)

­­

对于2d空间，我们需要切四次，3D切6次

先切top clip,再切bot clip（这里没超过边界），再切right clip,再切left clip(这里没超过边界)



­

将vertex array压缩的过程

首先第一个vertex，在长方形内保留，不然就舍去 //basic step

然后循环

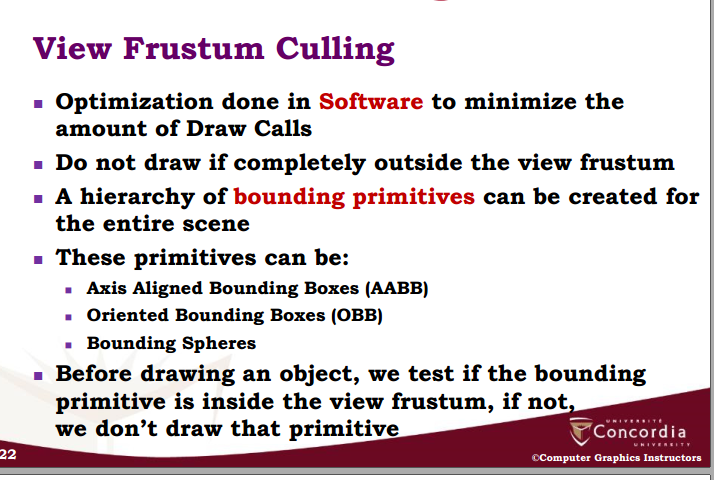
将上一个vertex与这个vertex比较，

Into in 保留第二个点

In to out，保留交点

Out to in，保留交点与里面的点

Out to out不保留



View frustum culling

视野剔除

Optimization 优先

最好在软件范围内实施来减少draw call的数量

如果完全在视野范围外，就不要画

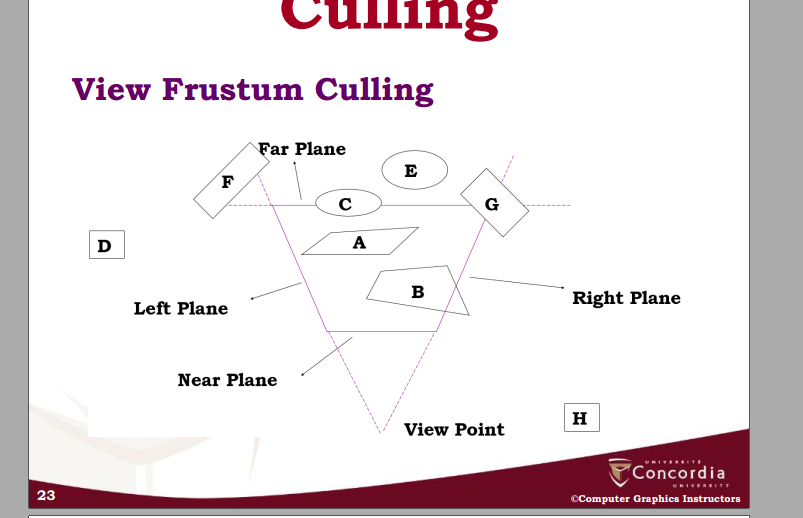
Entire Scene 可以由一些bounding primitives组成(用边框来给每一个object划定一个大致范围)

1.axis aligned bounding boxes AABB 轴向对齐边框

2.oriented bounding boxes，以原点为中心的边框

3.Bounding sphere 包围球

在画一个Object之前，我们测试场景的bounding primitive，那些不在视野范围内的bounding primitive就不会画这个primitive



可以看见我们给每一个物体都画了一个大致范围