20181001 그래픽스

Transformation을 할때는 각각의 object마다 model coordinate을 지정한 다음 그것에 변환을적용하면 된다

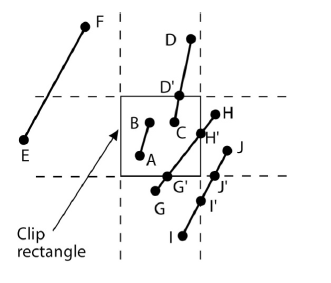
Viewport는 디바이스 화면 어디에 출력이 될지 결정하는 것이고 실제로는 window를 가지고 camera work를 결정한다!!

clipping이란 object가 화면에 표시될 때 화면에 띄워줄걸 투영시키는 과정 같은 느낌이다.

적당히 이상하게 생긴 polygon이 있을 경우 우린 그 vertex 정도와 line정보만 가지고있지만 실제로 띄워줘야 하는 건 그 내부의 색깔일 것이다. 따라서 vertex가 window밖에 있으면 헷갈린다. 따라서 이런 걸 handling해줄수 있어야 하는데 평범한 point같은 경우에는 x,y좌표가 window x\_min, x\_max, y\_min, y\_max 내부에 있는지 확인만 해주면 된다!

line같은 경우에는 일단 전부 포함하거나 전부 포함되지 않는 경우(trivially accepted, trivially rejected) 먼저 간편하게 처리해주고 line이 window의 내부와 외부에 모두 걸치는 경우에는 Cohen-Surtherland algorithm을 사용해준다.

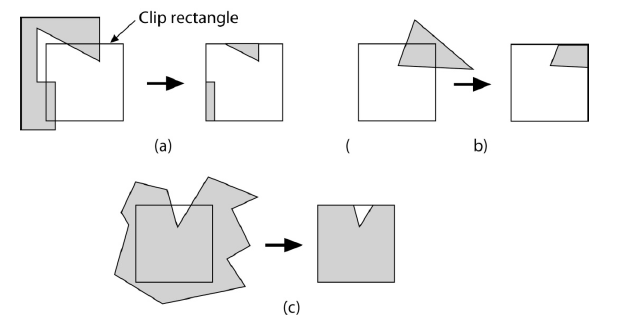
pdf에서 code는 윈도우를 기준으로 3x3으로 일단 나눈다



그런 다음 각각의 cell에 4digit Boolean code를 부여하는데 각각의 digit은 1행, 3행, 1열, 3열에 포함되어 있는 여부를 나타낸다. 따라서 1,1 cell의 code는 1010이 되는 것이며 2,2 cell은 0000, 3,2 cell의 경우 0100이 되는 것이다.

따라서 이런 line의 끝점 p0,p1을 생각한다면 code(p0) and code(p1)를 binary operation으로 진행한다면 같은 줄에 있을 때 true로 나오며 이것은 결국 앞에서 말했던 trivially rejected의 조건이 된다. 또한 code(p0) or code(p1)이 false로 나오면 code(p0)=code(p1)=0이란 의미이므로 둘다 trivially accepted 되어야 한다. 만약 이 두가지가 아니라면 그냥 line을 여러 조각으로 나누어 줘야 한다! 정확하게는 line을 2개로 나눈다음 line set에 포함 시켜야 한다.

근데 line이나 point까진 그렇다쳐도 polygon은 도대체 어떻게 해야할까??



이런 모양들이 나오면 생각만 해도 끔찍하다.

일단 이런 경우 polygon의 bounding box를 생각한다. Polygon을 정확하게 감싸는 rectangle을 생각한다. 그런 다음 이런 bounding box가 window에 정확하게 포함되냐 아니면 아얘 밖이냐에 따라 trivially accepted와 rejected 케이스를 먼저 분류해준다. 그런 다음 문제는 또 걸치는 케이스가 문제이다. 근데 이건 굉장히 복잡해보인다! 방법에 대해선 수업하지 않는 것 같다.