컴퓨터 그래픽스 HW4

제출 기한: 12월 21일 까지

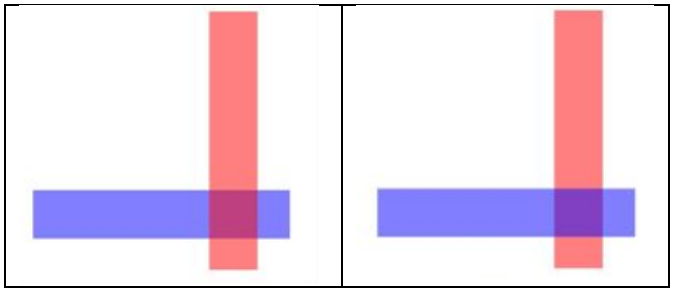
1. 코드 실행 결과를 볼 수 있는 실행 화면 캡쳐, 어떻게 코드를 작성했는지 이에 대한 설명 및 토의 사항을 적은 리포트 (pdf 포맷)와 실행 코드 (.cpp)들을 하나의 파일로 압축하여 제출
2. 가시 공간 (부피), 물체, polyline의 색은 필요한 경우 적절히 선택하고 리포트에 간단히 설명하자.

1. Lecture 15 page 37-38에 있는 basic barn 메쉬 정보를 이용하여 basic barn 메쉬를 그리고자 한다. gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)을 이용하여 입체감 있게 LOS를 조정하자. 조명 모델을 사용하지 않고 각 face에는 yellow color를 입혀보고 투영 시 'glFrustum'을 사용하자. 배경색은 black color로 정하자.

2. 이번에는 1의 basic barn의 각 face에 있는 실제 bitmap image의 texture를 입히려고 한다. 이 경우에 1에서 사용한 face의 color는 무시하고 첨부한 'canTop.bmp' 이미지 파일을 texture로 사용하자. 수동 texture mapping 시에 왜곡이 생기지 않도록 basic barn의 face 모양 (비율)과 texture space에서의 모양 (비율)을 동일하도록 수동 texture mapping을 수행하자. Texture에서 어느 부분을 가져올지는 여러분이 직접 정하고 리포트에 어떻게 왜곡이 없도록 수동 texture mapping을 수행했는지 설명하자.

3. 이번에는 2의 결과에 Lecture 14 page 52의 예에서 사용한 조명 모델을 사용해 보자. 'glNormal3f' 함수를 사용하고 정규화된 법선벡터를 사용하자. Lecture 18와 같이 texture 옵션 설정 시에 'GL_REPLACE'와 'GL_MODULATE'에 차이가 있는지 확인해보고 두 결과를 모두 리포트에 첨부해보자. 'GL_MODULATE'를 사용시에는 1에서 사용한 face의 색과 texture의 색이 혼합되어 나오는가?

4. 아래 왼쪽 오른쪽 그림은 두 개의 rectangle과 수업시간에 배운 alpha값을 이용한 blending 방법을 사용하여 그린 것이다. 아래에서 배경은 white, 두 rectangle의 색은 각각 red와 blue이다. Alpha값을 얼마를 주면 아래와 같은 결과를 얻을 수 있을까? 왼쪽 오른쪽 그림에서 각각 source와 destination은 무엇일까? 리포트에 설명해 보자. 왜 아래와 같이 보이는지 리포트에 설명해 보자. 만일 glEnable(GL_DEPTH_TEST)를 사용하여 depth buffer를 사용하면 결과가 어떻게 바뀔까? 출력 결과를 보여주고 결과에 대해서 이유를 설명해보자.



5. 아래와 같이 구 (glutSolidSphere)와 polygon을 정의하고 타이머 콜백 함수를 사용하여 구가 그림 (b)와 같이 평면을 굴러서 내려오는 것 같은 애니메이션을 만들어보자. 이를 위해 투영, 카메라의 위치 및 LOS를 조절하자 (단, 완벽하게 아래와 같이 맞추지는 않아도 된다). 시간에 따라서 아래로 내려갈수록 더 빨리 떨어지는 것처럼 보이게 식을 조절하자 (glTranslatef(); 사용시). 캡쳐시 구의 움직임을 볼 수 있도록 시간에 따라서 여러 번 캡쳐해 보자. 조명 처리를 위하여 Lecture 14 page 52의 예에서 사용한 조명 모델을 사용해 보자.

6. Lecture 17 page 9에 있는 2차 스플라인 함수, g(t), 를 이용 아래 그림과 같은 curve를 그리고자 한다. 아래 그림을 보면 총 7개의 control point ($P_0, \ldots, P_6$)를 사용하였다. 이 curve는 수식 $V(t) = \sum_{k=0}^{6} P_k g(t-k), 2 \le t \le 7$, 를 이용하면 그릴 수 있다. 여기서 P_k는 k번째 control point이다. 2차 스플라인 함수 수식은 강의 자료를 이용하고 Control point의 위치는 아래 그림을 이용하여 정하자. 코드 작성시에 반복문을 사용하여 t값을 2에서부터 7까지 조금씩 증가시키면서 'GL_LINE_STRIP' 함수를 이용하여 아래 그림처럼 curve를 그려보자. Curve를 그린 결과를 출력하고 아래 그림처럼 t=2, 3,..., 7에서는 두 control point의 중점에 위치하는 것을 확인하고 이를 실행창에 출력한 결과를 리포트에 넣자.