HW 1 Report

Yêu Cầu

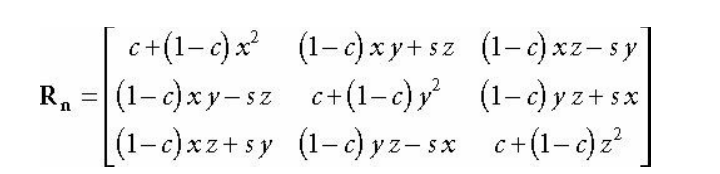
HW1 is about just filling in four functions in the given code to do two things: turn a teapot image horizontally and vertically. Well, actually this is what it looks like, we are in fact moving our camera: see the "crystall ball interface" explanation in the first page (Introduction) of the homework.

Four functions for two things? Yes. Transform::left and Transform::up are called to do the deeds, but they should refer to and use Transform::rotate to do it. And the fourth function Transform::lookat is just for us to learn how to play with matrices and vectors of geometry to perform the lookat function. In fact, I think you may actually start working on it independently from the other three functions but it is better to leave it to the end.

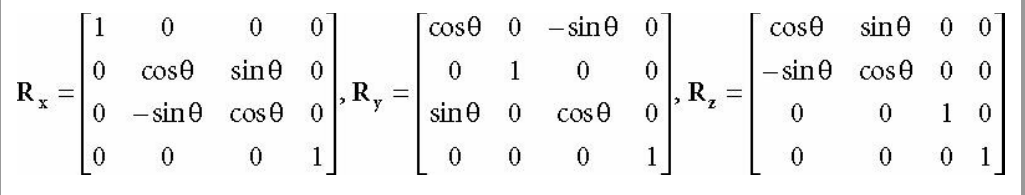
All your code could take up to 10 to 20 lines of code really, so do not despair. The tough part is understanding, not coding.

Vài điều rút ra sau khi hoàn thành HW1

1/ Như đề bài đã nói, phần khó nhất ở đây không phải là code, mà là hiểu vấn đề. Ở những bước đầu tiên, đề bài yêu cầu chúng ta viết 2 hàm để xoay ấm trà theo chiều dọc và chiều ngang. Quay lại với kiến thức được dạy ở phần này, chúng ta được biết có 2 cách để xác định “Rotation Matrix” của vật:



Hoặc



Cách đầu tiên có tên gọi là axis-angle, cách thứ 2 được suy ra từ việc xoay vật ở không gian 2D. Ở đây chúng ta ( hoặc em :v ) có xu hướng sẽ thích dùng cách 2 hơn vì nó dễ nhớ và dễ làm. Tuy nhiên nhìn lại các parameters mà 2 hàm chúng ta được cung cấp như sau:  




Chúng ta có góc quay (degrees), tọa độ của camera (eye) và vector chỉ hướng lên (up). Ở đây họ không cho position của vật, nên dẫn tới chúng ta sẽ nghĩ theo hướng là xoay camera do chúng ta có position của camera. Thực tế thì nếu chúng ta làm cách 2, cần phải tính thêm new position của camera sau khi xoay, chứ không phải chỉ cần nhân với Rotation Matrix là được. Lỗi này với em thì là 1 critical bug vì khi chỉ thử đơn lẻ từng hàm Up/Left sẽ không thể phát hiện ra được, vì lúc này cho kết quả đúng. Nhưng khi kết hợp thêm nhiều phép xoay khác nhau sẽ cho kết quả sai.

* Theo em nghĩ, nếu như trong trường hợp cần xoay vật thì nên áp dụng cách 2, còn nếu cần xoay camera thì nên áp dụng theo cách 1.

2/ Khi đọc Solution, em để ý ở hàm Rotate, sau khi tính toán ra ma trận M, họ sẽ return ma trận MT. Và ở các hàm tính toán khác, thay vì cho

(ma trận vị trí mới của vật) = (ma trận vị trí của vật) \* (ma trận biến đổi M)

thì họ lại để là

(ma trận vị trí mới của vật) = (ma trận biến đổi M) \* (ma trận vị trí của vật)

Sau khi tìm hiểu thì em được biết là ở giai đoạn Shader làm việc, các ma trận sẽ được chuyển thành ma trận chuyển vị của chính nó. Nguyên nhân của việc này là do thiết kế phần cứng, thay vì dùng các ô nhớ liền kề nhau tạo thành 1 hàng của ma trận thì lại chuyển thành cột ?? :D ?? . Trong DirectX, việc chuyển đổi này được compiler thực hiện tự động nên chúng ta không cần phải tính ma trận chuyển vị mà cứ lắp theo đúng công thức là được. Tuy nhiên OpenGL không hỗ trợ việc này nên chúng ta phải tự làm. Sử dụng tính chất chuyển vị của ma trận như sau:

P’ = P \* M = (MT \* PT )T

\*Ma trận P ở đây thường sẽ là 1 ma trận 3x1 (hoặc 4x1 trong homogenous coordinates).

Link tham khảo: http://www.mindcontrol.org/~hplus/graphics/matrix-layout.html?fbclid=IwAR3vlD1SA5X29sUpd6wYCRTCaw3Dbe4POQxUntj57\_CpicfPqbfr\_nldbRo