	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación


Laboratorio de Computación Gráfica e Interacción Humano Computadora

Modelado Jerárquico

N° de practica: 04

Nombre completo de los alumnos		Firma
Arroyo Llanes Miguel Alejandro		
N° de brigada: 2	Fecha de ejecución: 08/03/2025	Grupo: 03
Calificación:	Profesor: Ing. Jose Roque Roman Guadarrama	

Elaborado por:	Revisado por:	Autorizado por:	Vigente desde:
-----------------------	----------------------	------------------------	-----------------------

	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

Comentario:

Primeramente, se completó la figura la de la grúa, esto se hizo creando una pirámide rectangular en la parte inferior del cuerpo de la base, luego se colocaron cilindros pequeños para simular las ruedas y se les agrego movimiento, esto se logro editando los archivos window para poder agregar los movimientos con las teclas “u”, “i”, “o” y “p”.

Posteriormente se hizo el dibujo de un perro robótico con articulaciones en las patas (independiente las 2 delanteras y las 2 traseras), además de el movimiento de la cola con las teclas “j”, “k” y “l”.

Bloque de código:

Archivo window.h agregando los nuevos movimientos.

```

38 ~Window();
39 private:
40   GLFWwindow *mainWindow;
41   GLint width, height;
42   GLfloat rotax, rotay, rotaz, articulacion1, articulacion2, articulacion3, articulacion4, articulacion5, articulacion6;
43   GLfloat articulacion7, articulacion8, articulacion9, articulacion10; //agregar estas variables
44
45


```

Archivo window.cpp agregando los nuevos movimientos.

```

Window::Window(GLint windowHeight, GLint windowHeight)
{
    width = windowHeight;
    height = windowHeight;
    rotax = 0.0f;
    rotay = 0.0f;
    rotaz = 0.0f;
    articulacion1 = 0.0f;
    articulacion2 = 0.0f;
    articulacion3 = 0.0f;
    articulacion4 = 0.0f;
    articulacion5 = 0.0f;
    articulacion6 = 0.0f;
    articulacion7 = 0.0f;
    articulacion8 = 0.0f;
    articulacion9 = 0.0f;
    articulacion10 = 0.0f;
}

```

	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

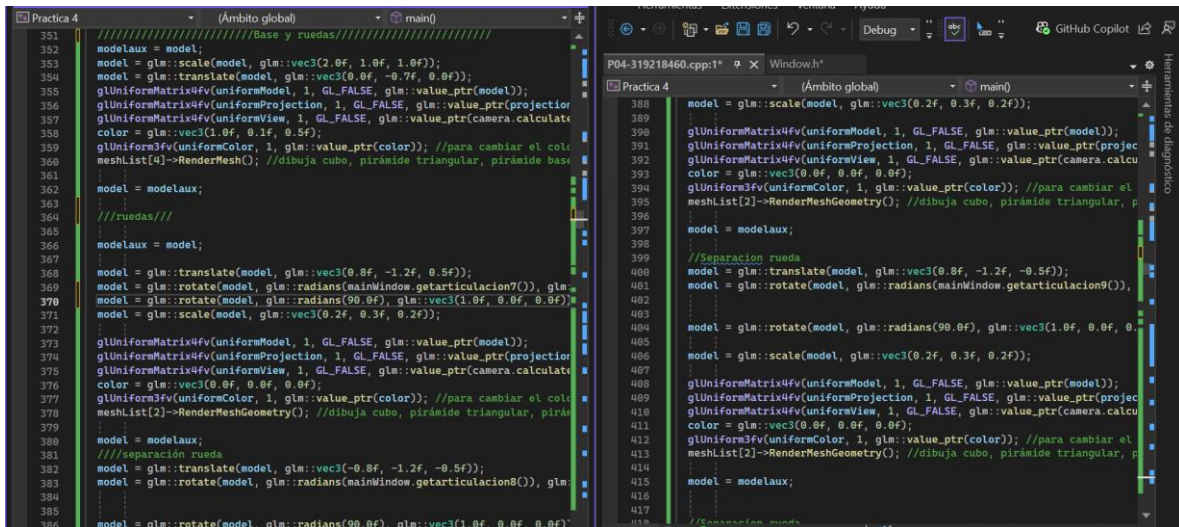
```


///Asignación para las 4 ruedas///

if (key == GLFW_KEY_U)
{
    theWindow->articulacion7 += 10.0;
}
if (key == GLFW_KEY_I)
{
    theWindow->articulacion8 += 10.0;
}
if (key == GLFW_KEY_O)
{
    theWindow->articulacion9 += 10.0;
}
if (key == GLFW_KEY_P)
{
    theWindow->articulacion10 += 10.0;
}

```

Dibujado de la base de la grúa y sus ruedas.



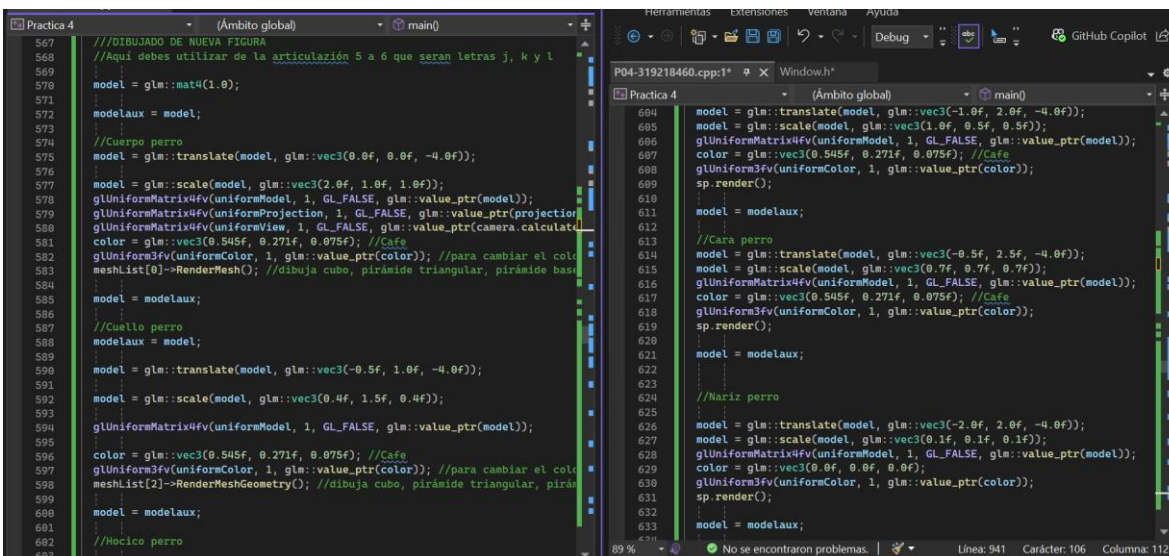
	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

```

418 //Separación rueda
419
420 model = glm::translate(model, glm::vec3(-0.8f, -1.2f, 0.5f));
421 model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10()), glm::vec3(1.0f, 0.0f, 0.0f));
422
423
424 model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
425
426 model = glm::scale(model, glm::vec3(0.2f, 0.3f, 0.2f));
427
428 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
429 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
430 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView()));
431 color = glm::vec3(0.0f, 0.0f, 0.0f);
432 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
433 meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide cuadrada
434
435 model = modelaux;

```


Dibujado perro robot.



```

567 //DIBUJADO DE NUEVA FIGURA
568 //Aquí debes utilizar de la articulación 5 a 6 que serán letras j, k y l
569
570 model = glm::mat4(1.0);
571
572 modelaux = model;
573
574 //Cuerpo perro
575 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
576
577 model = glm::scale(model, glm::vec3(2.0f, 1.0f, 1.0f));
578 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
579 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
580 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView()));
581 color = glm::vec3(0.545f, 0.271f, 0.075f); //Cafe
582 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
583 meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide cuadrada
584
585 model = modelaux;
586
587 //Cuello perro
588 modelaux = model;
589
590 model = glm::translate(model, glm::vec3(-0.5f, 1.0f, -4.0f));
591
592 model = glm::scale(model, glm::vec3(0.4f, 1.5f, 0.4f));
593
594 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
595
596 color = glm::vec3(0.545f, 0.271f, 0.075f); //Cafe
597 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
598 meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide cuadrada
599
600 model = modelaux;
601
602 //Hecio perro
603
604
605
606
607
608
609
610
611
612
613 //Cara perro
614 model = glm::translate(model, glm::vec3(-0.5f, 2.5f, -4.0f));
615 model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
616 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
617 color = glm::vec3(0.545f, 0.271f, 0.075f); //Cafe
618 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
619 sp.render();
620
621
622
623
624 //Nariz perro
625
626 model = glm::translate(model, glm::vec3(-2.0f, 2.0f, -4.0f));
627 model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
628 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
629 color = glm::vec3(0.0f, 0.0f, 0.0f);
630 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
631 sp.render();
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

	<h1>Manual de Prácticas</h1>
<p>Secretaría/División: División de Ingeniería Eléctrica</p>	<p>Área/Departamento: Ingeniería en Computación</p>

Practica 4

(Ámbito global)

main()

```

635 //Lengua metálica del perro robot
636
637 model = glm::translate(model, glm::vec3(-1.0f, 1.6f, -4.0f));
638 model = glm::rotate(model, glm::radians(45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
639 model = glm::scale(model, glm::vec3(0.3f, 0.1f, 0.1f));
640 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
641 color = glm::vec3(0.75f, 0.75f, 0.75f);
642 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
643 sp.render();
644
645 model = modelaux;
646
647 //Ojo robotico
648 model = glm::translate(model, glm::vec3(-1.0f, 2.5f, -4.0f));
649
650 model = glm::scale(model, glm::vec3(0.4f, 0.3f, 0.7f));
651 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
652 color = glm::vec3(0.75f, 0.75f, 0.75f);
653 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
654 sp.render();
655
656 model = modelaux;
657
658 //Oreja derecha
659 model = glm::translate(model, glm::vec3(-0.5f, 3.0f, -4.3f));
660 model = glm::rotate(model, glm::radians(-30.0f), glm::vec3(1.0f, 0.0f, 0.0f));
661
662 model = glm::scale(model, glm::vec3(0.3f, 0.8f, 0.3f));
663 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
664 color = glm::vec3(0.545f, 0.271f, 0.075f);
665 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
666 sp.render();
667
668 model = modelaux;
669
670 //Oreja izquierda
671 model = glm::translate(model, glm::vec3(-0.5f, 3.0f, -3.7f));

```

P04-319218460.cpp:1*

Window.h*

Practica 4

(Ámbito global)

main()

```

672
673 model = glm::rotate(model, glm::radians(30.0f), glm::vec3(1.0f, 0.0f, 0.0f));
674
675 model = glm::scale(model, glm::vec3(0.3f, 0.8f, 0.3f));
676 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
677 color = glm::vec3(0.545f, 0.271f, 0.075f);
678 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
679 sp.render();
680
681 model = modelaux;
682
683 //Pata delantera izquierda
684 //base
685 glm::mat4 modelLeft = glm::translate(glm::mat4(1.0f), glm::vec3(-0.5f, -
686
687 modelLeft = glm::rotate(modelLeft, glm::radians(mainWindow.getarticulaci
688 glm::mat4 modelaux = modelLeft;
689
690
691 modelLeft = glm::scale(modelLeft, glm::vec3(0.3f, 0.3f, 0.3f));
692 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelLeft));
693 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projec
694 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calcu
695 color = glm::vec3(0.75f, 0.75f, 0.75f);
696 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
697 sp.render();
698
699
700
701 //Tuerca

```

Practica 4

(Ámbito global)

main()

```

699 //Tuerca
700
701 modelLeft = glm::translate(modelLeft, glm::vec3(0.0f, 0.0f, 1.0f));
702 modelaux = modelLeft;
703 modelLeft = glm::scale(modelLeft, glm::vec3(0.3f, 0.3f, 0.3f));
704 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelLeft));
705 color = glm::vec3(0.0f, 0.0f, 0.0f);
706 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
707 sp.render();
708
709 modelLeft = modelaux;
710
711
712
713 //Pierna
714
715 modelLeft = glm::translate(modelLeft, glm::vec3(0.0f, -1.0f, -1.0f));
716 modelaux = modelLeft;
717
718 modelLeft = glm::scale(modelLeft, glm::vec3(0.0f, 1.5f, 0.0f));
719
720 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelLeft));
721 color = glm::vec3(0.75f, 0.75f, 0.75f);
722 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar
723 meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular
724
725 modelLeft = modelaux;
726
727 //Pie
728
729 modelLeft = glm::rotate(modelLeft, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
730 modelLeft = glm::translate(modelLeft, glm::vec3(-1.0f, 0.5f, 0.0f));
731 modelLeft = glm::scale(modelLeft, glm::vec3(1.5f, 2.5f, 1.5f));
732
733 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelLeft));
734 color = glm::vec3(0.75f, 0.75f, 0.75f);
735 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar

```

P04-319218460.cpp:1*

Window.h*

Practica 4


(Ámbito global)

main()

```

736 meshList[4]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, p
737
738 // Restaurar modelo antes de pasar a la siguiente pata
739 modelLeft = modelaux;
740
741 //Pata delantera derecha
742 //base
743 glm::mat4 modelRight = glm::translate(glm::mat4(1.0f), glm::vec3(-0.5f, -
744
745 modelRight = glm::rotate(modelRight, glm::radians(mainWindow.getarticulaci
746
747 modelaux = modelRight;
748
749
750 modelRight = glm::scale(modelRight, glm::vec3(0.3f, 0.3f, 0.3f));
751 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelRight));
752 glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projec
753 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calcu
754 color = glm::vec3(0.75f, 0.75f, 0.75f);
755 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
756 sp.render();
757
758
759 modelRight = modelaux;
760
761 //Tuerca
762
763 modelRight = glm::translate(modelRight, glm::vec3(0.0f, 0.0f, -0.3f));
764 modelaux = modelRight;
765
766 modelRight = glm::scale(modelRight, glm::vec3(0.1f, 0.1f, 0.1f));
767 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelRight));
768

```


	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

P04-319218460.cpp:2*

Practica 4

(Ámbito global)

main()

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

```

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelRight));
color = glm::vec3(0.f, 0.f, 0.f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

modelRight = modelaux;

//Pierna
modelRight = glm::translate(modelRight, glm::vec3(0.0f, -0.3f, 0.4f));
modelaux = modelRight;

modelRight = glm::scale(modelRight, glm::vec3(0.25f, 0.5f, 0.25f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelRight));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide
modelRight = modelaux;

//Pie
modelRight = glm::rotate(modelRight, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelRight = glm::scale(modelRight, glm::vec3(0.5f, 0.7f, 0.5f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelRight));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
meshList[4]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide
// Restaurar modelo antes de pasar a la siguiente pata
modelRight = modelaux;

```

P04-319218460.cpp:1*

Practica 4

(Ámbito global)

main()

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

```

// Pata trasera izquierda
// base
glm::mat4 modelBackLeft = glm::translate(glm::mat4(1.0f), glm::vec3(0.5f, 0.0f, 0.0f));
modelBackLeft = glm::rotate(modelBackLeft, glm::radians(mainWindow.getAspectRatio() * 0.5f), glm::vec3(0.0f, 0.0f, 1.0f));
glm::mat4 modelaux = modelBackLeft;

modelBackLeft = glm::scale(modelBackLeft, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackLeft));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView()));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Turca
modelBackLeft = glm::translate(modelBackLeft, glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = modelBackLeft;
modelBackLeft = glm::scale(modelBackLeft, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackLeft));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

modelBackLeft = modelaux;

// Pierna
modelBackLeft = glm::translate(modelBackLeft, glm::vec3(0.0f, -1.0f, -1.0f));
modelaux = modelBackLeft;

modelBackLeft = glm::scale(modelBackLeft, glm::vec3(0.0f, 1.5f, 0.8f));

```

Practica 4

(Ámbito global)

main()

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

```

modelBackLeft = glm::scale(modelBackLeft, glm::vec3(0.8f, 1.5f, 0.8f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackLeft));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide
modelBackLeft = modelaux;

// Pie
modelBackLeft = glm::rotate(modelBackLeft, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelBackLeft = glm::translate(modelBackLeft, glm::vec3(-1.0f, 0.5f, 0.0f));
modelBackLeft = glm::scale(modelBackLeft, glm::vec3(1.5f, 2.5f, 1.5f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackLeft));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
meshList[4]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide
// Restaurar modelo antes de pasar a la siguiente pata
modelBackLeft = modelaux;

// Pata trasera derecha
// base
glm::mat4 modelBackRight = glm::translate(glm::mat4(1.0f), glm::vec3(0.5f, 0.0f, 0.0f));
modelBackRight = glm::rotate(modelBackRight, glm::radians(mainWindow.getAspectRatio() * 0.5f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = modelBackRight;

modelBackRight = glm::scale(modelBackRight, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackRight));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateView()));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

```

P04-319218460.cpp:1*

Practica 4

(Ámbito global)

main()

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

```

sp.render();

modelBackRight = modelaux;

// Turca
modelBackRight = glm::translate(modelBackRight, glm::vec3(0.0f, 0.0f, -0.5f));
modelaux = modelBackRight;
modelBackRight = glm::scale(modelBackRight, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackRight));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

modelBackRight = modelaux;


// Pierna
modelBackRight = glm::translate(modelBackRight, glm::vec3(0.0f, -0.3f, 0.4f));
modelaux = modelBackRight;

modelBackRight = glm::scale(modelBackRight, glm::vec3(0.25f, 0.5f, 0.25f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackRight));
color = glm::vec3(0.75f, 0.75f, 0.75f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide
modelBackRight = modelaux;

// Pie
modelBackRight = glm::rotate(modelBackRight, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelBackRight = glm::scale(modelBackRight, glm::vec3(0.5f, 0.7f, 0.5f));

```

	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

P04-319218460.cpp2*

Practica 4

(Ámbito global)

main()

```

901 modelBackRight = glm::translate(modelBackRight, glm::vec3(-0.4f, 0.2f, 0.0f));
902 modelBackRight = glm::scale(modelBackRight, glm::vec3(0.5f, 0.7f, 0.5f));
903
904 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelBackRight));
905 color = glm::vec3(0.75f, 0.75f, 0.75f);
906 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color
907 meshList[4] -> RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide
908
909 // Restaurar modelo antes de pasar a la siguiente pata
910 modelBackRight = modelAux;
911
912
913 // Base de la cola
914 glm::mat4 modelTail = glm::translate(glm::mat4(1.0f), glm::vec3(1.0f, 0.0f, 0.0f));
915 modelTail = glm::rotate(modelTail, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
916 glm::mat4 modelAux1 = modelTail;
917
918
919 modelTail = glm::scale(modelTail, glm::vec3(0.2f, 1.5f, 0.2f));
920 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelTail));
921 color = glm::vec3(0.545f, 0.271f, 0.875f);
922 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
923 meshList[2] -> RenderMeshGeometry();
924
925 modelTail = modelAux1;
926
927 //Articulación cola
928 modelTail = glm::translate(modelTail, glm::vec3(0.0f, -1.0f, -0.0f));
929 modelTail = glm::rotate(modelTail, glm::radians(mainWindow.getarticulacion6));
930
931 modelTail = glm::scale(modelTail, glm::vec3(0.3f, 0.3f, 0.3f));
932 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelTail));
933 color = glm::vec3(0.75f, 0.75f, 0.75f);
934 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
935 sp.render();
936 glm::mat4 modelAux2 = modelTail;
937
938 //Punta cola
939
940 modelTail = glm::translate(modelTail, glm::vec3(1.4f, -0.9f, 0.0f));
941 modelTail = glm::rotate(modelTail, glm::radians(-120.0f), glm::vec3(0.0f, 0.0f, 1.0f));
942 modelTail = glm::scale(modelTail, glm::vec3(0.5f, 3.0f, 0.5f));
943 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelTail));
944 color = glm::vec3(0.545f, 0.271f, 0.875f);
945 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
946 meshList[3] -> RenderMeshGeometry();
947 modelTail = modelAux2;
948

```

P04-319218460.cpp1*

Practica 4

(Ámbito global)

main()

```

918 modelTail = glm::scale(modelTail, glm::vec3(0.2f, 1.5f, 0.2f));
919 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelTail));
920 color = glm::vec3(0.545f, 0.271f, 0.875f);
921 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
922 meshList[2] -> RenderMeshGeometry();
923
924 modelTail = modelAux1;
925
926 //Articulación cola
927 modelTail = glm::translate(modelTail, glm::vec3(0.0f, -1.0f, -0.0f));
928 modelTail = glm::rotate(modelTail, glm::radians(mainWindow.getarticulacion6));
929
930 modelTail = glm::scale(modelTail, glm::vec3(0.3f, 0.3f, 0.3f));
931 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelTail));
932 color = glm::vec3(0.75f, 0.75f, 0.75f);
933 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
934 sp.render();
935 glm::mat4 modelAux2 = modelTail;
936
937 //Punta cola
938
939 modelTail = glm::translate(modelTail, glm::vec3(1.4f, -0.9f, 0.0f));
940 modelTail = glm::rotate(modelTail, glm::radians(-120.0f), glm::vec3(0.0f, 0.0f, 1.0f));
941 modelTail = glm::scale(modelTail, glm::vec3(0.5f, 3.0f, 0.5f));
942 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelTail));
943 color = glm::vec3(0.545f, 0.271f, 0.875f);
944 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
945 meshList[3] -> RenderMeshGeometry();
946 modelTail = modelAux2;
947

```

89 %


No se encontraron problemas.

Línea: 951

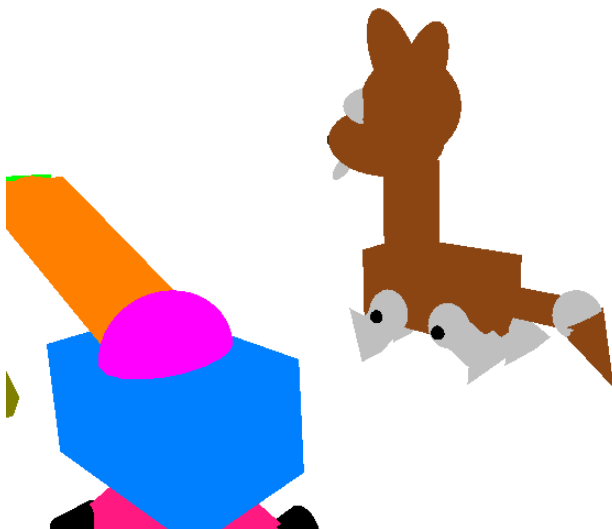
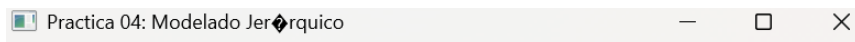
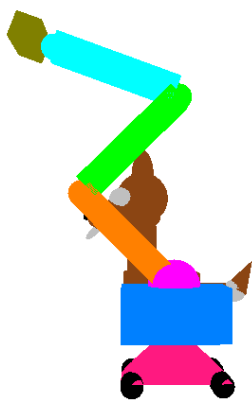
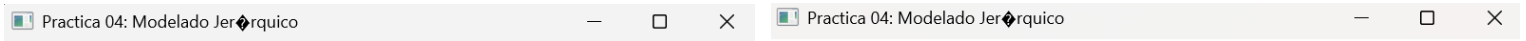
Carácter: 3


Columna: 9

M

	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

Ejecución:



	Manual de Prácticas
Secretaría/División: División de Ingeniería Eléctrica	Área/Departamento: Ingeniería en Computación

Conclusión:

Se han comprendido los conceptos del modelado, así como cada vez se acerca más un dominio de la traslación, rotación y manipulación de dimensiones de una figura, así como el concepto del modelado jerárquico.

El desarrollo de esta practica aunque reconozco que no muy difícil si fue muy laboriosa, aun se debe de buscar algún método para que el desarrollo de este tipo de programas sea más sencillo y menos laborioso.

Bibliografía:

1. No se buscó en ninguna fuente externa.