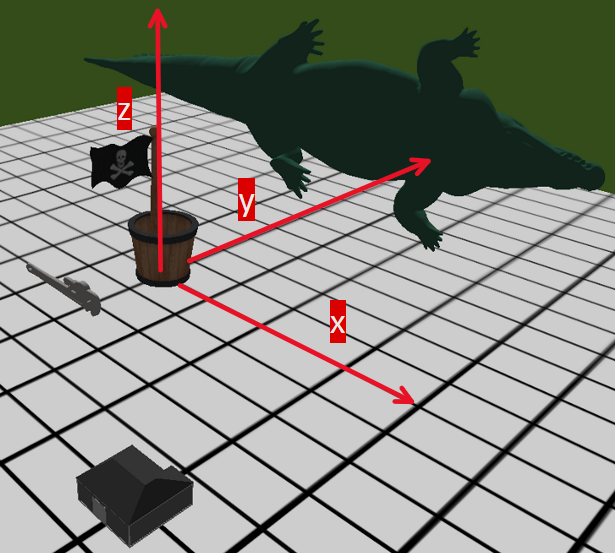
Graphics Project – COSC3000 2024 S1

*By Joy Yin – s4740051*

**DISPLAYING 3D OBJECTS WITH TRANSFORMATIONS:**

* Python was used to display the objects and perform transformations.
* Added a flag object that will not be transformed from the starting position as a ***reference point*** to demonstrate the transformations performed on other objects.
* The ‘ground’ in this world is represented with the grid.

Beginning Position:



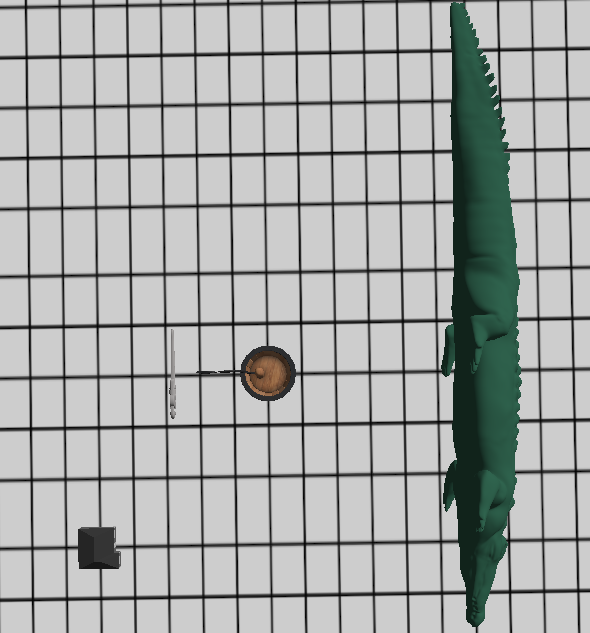
****

Figure 2: Beginning position with axes – perspective one.

Figure 1: Beginning position Bird’s Eye View

A frog on a tile floor

Description automatically generated

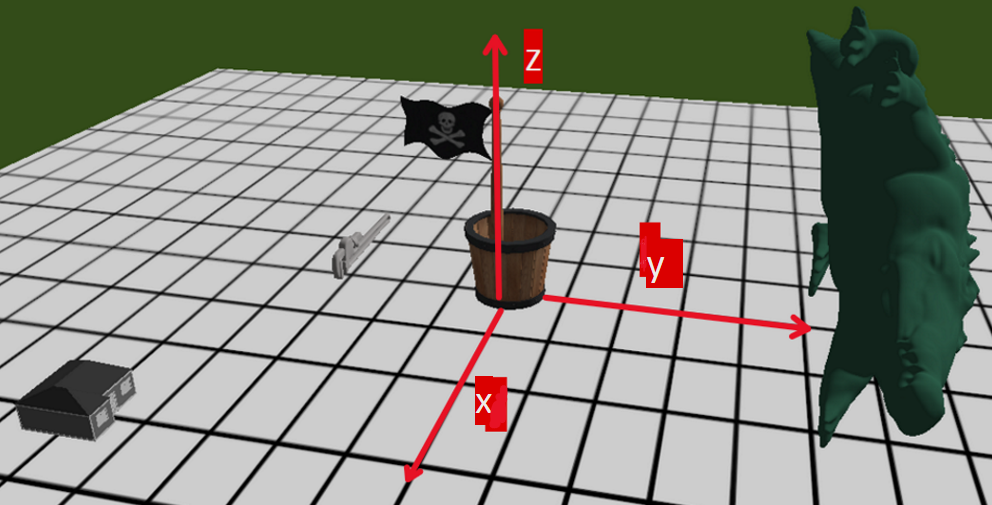
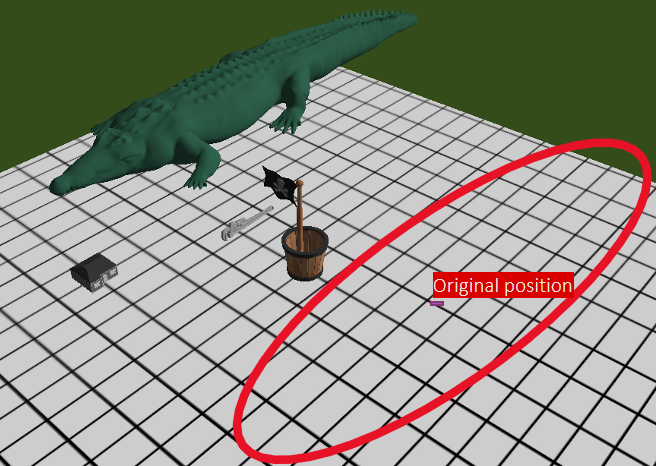


Figure 4: Beginning position from low perspective.

Figure 3: Beginning position with axes – perspective two.

Rotating Crocodile:

* The first attempt used the make\_rotation\_x() function but this rotated the crocodile around the origin point (flag), instead of rotating in its place.
* As such, a new transformation function make\_rotation\_around\_point() was created to rotate the object around a specified point.
* The crocodile was then rotated 90 degrees about the x-axis .

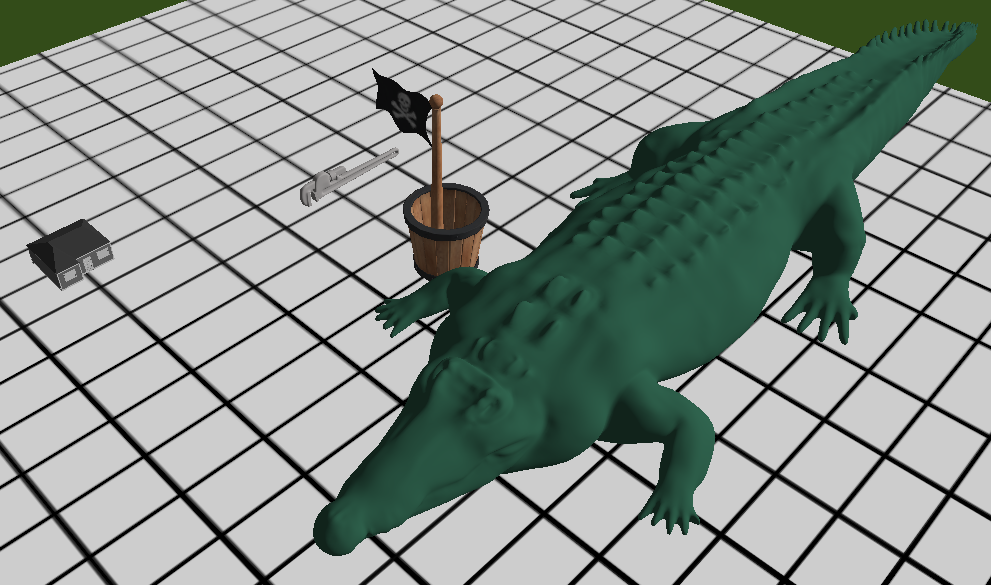


A screen shot of a computer program

Description automatically generated

Figure 6: Function code for rotating around point.

Figure 5: Crocodile rotation attempt without rotating around point, with original position marketed.

 A green alligator on a tile floor

Description automatically generated

Figure 8: Crocodile rotation, rotating around point – perspective two.

Figure 7: Crocodile rotation, rotating around point – perspective one.

A computer screen shot of a black screen

Description automatically generated

Figure 10: Code to rotate crocodile with the make\_rotation\_around\_point() function.

Translating the Wrench:

* The wrench was translated along the y-axis and x-axis, behind the flag post.

A green alligator on a white grid with a pirate flag and a bucket of wood

Description automatically generated

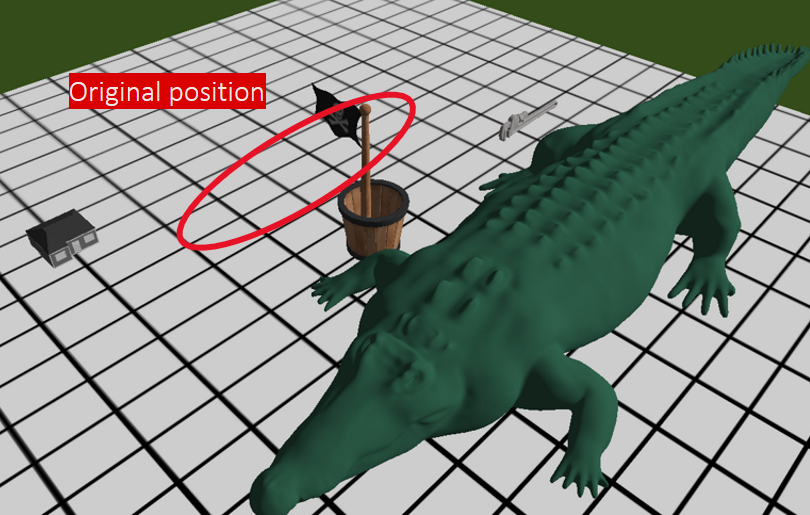


Figure 12: Translating the wrench – perspective two.

Figure 11: Translating the wrench with original position marked – perspective one.

Scaling and Rotating the House:

* Scaled up the house by a factor of 5.
* Rotated the house about z-axis using the new make\_rotation\_around\_point() function.

A green alligator on a tile floor

Description automatically generated

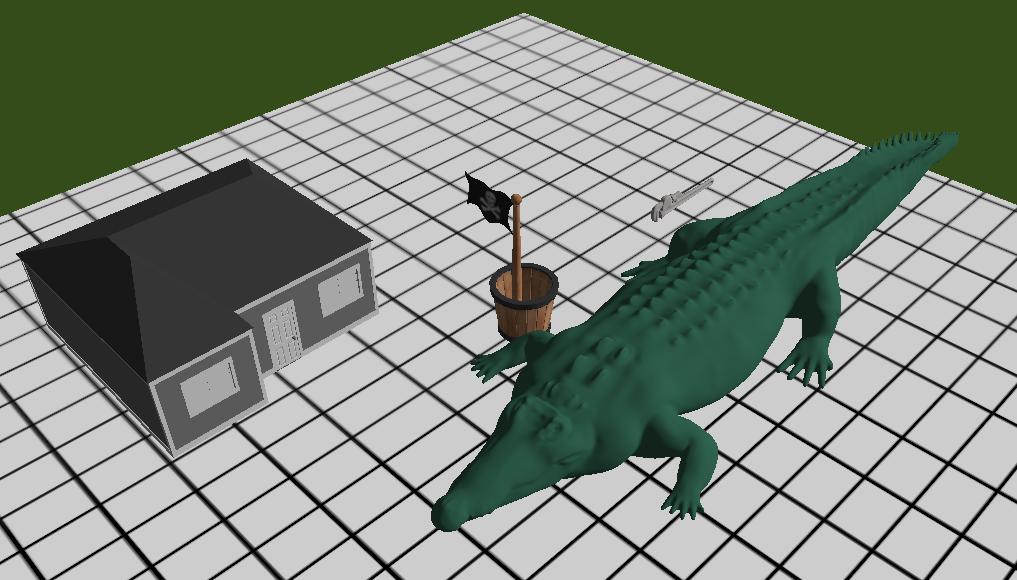


Figure 14: Rotated house about the z-axis, into final position.

Figure 13: Scaled house by a factor of 5.

**LIGHTING PATCHES:**

* To compute the scalar product and determine whether triangular patches were facing the light source, the following function was used:

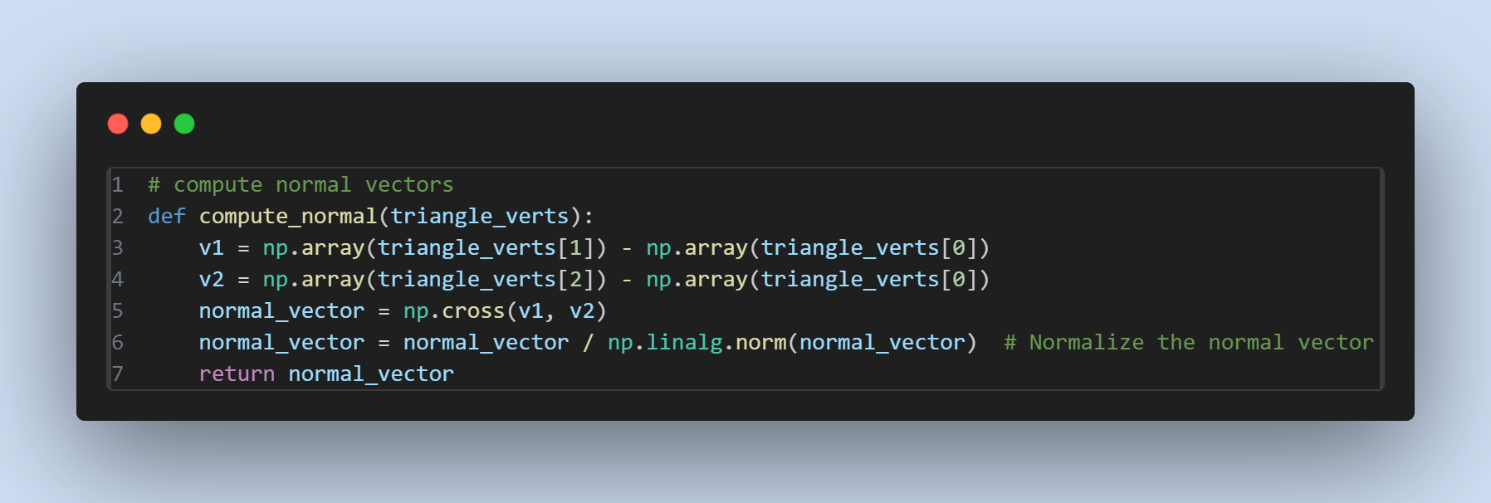
****

Figure 15: compute\_normal() function code.

* Four triangular patches were used to demonstrate different lighting effects, each one coloured differently show demonstrate the different facets.

A colorful pyramid shaped object

Description automatically generated

A colorful pyramid shaped object

Description automatically generated

A green yellow and blue diamond

Description automatically generated

Figure 18: Four triangle patches – below-right view.

Figure 17: Four triangle patches – left side view.

Figure 16: Four triangle patches – front on view.

* A camera object was then loaded in, rotated and translated into a position to represent the source of light. An extra triangle was drawn in to represent the light itself.

A camera with a yellow and green triangle

Description automatically generated

**A camera lens with a yellow light coming out of it

Description automatically generated**

Figure 20: Camera and light with triangle patches – right side view.

Figure 19: Camera and light with triangle patches – front on view.

A camera with a lens and a triangle

Description automatically generated with medium confidence

**A camera lens with a yellow light coming out of it

Description automatically generated**

Figure 22: Camera and light with triangle patches – top-left view.

Figure 21: Camera and light with triangle patches – below-right view.

* Triangles were then shaded according to their dot products.
* For all triangles with a negative dot product (facing away from light source), they were coloured completely black as it’s assumed that light cannot reflect off any surface in this ‘world’.

*Normal vector for triangle 1:*

*Dot Product: 1.962990915244728 (bright green)*

*Normal vector for triangle 2:*

*Dot Product: -0.5773502691896258 (black)*

*Normal vector for triangle 3:*

*Dot Product: -1.962990915244728 (black)*

*Normal vector for triangle 4:*

*Dot Product: 0.5773502691896258 (dark green)*

*Vector pointing towards the origin:*

* Given that there were two patches (triangles 1 & 4) facing the light source, the lower left triangle (triangle 4) receives less light than the one above it by a factor pf .
* As such, the colour intensity was decreased to 30%.

**A camera with a light coming out of it

Description automatically generated**

**A black flashlight with yellow light coming out of a green square

Description automatically generated**

Figure 23: Camera and light with lighting shaded triangle patches – front on view.

Figure 24: Camera and light with lighting shaded triangle patches – left view.

**A black camera with yellow light coming out of a black cube

Description automatically generated**

Figure 25: Camera and light with lighting shaded triangle patches – right view.