GAME2016
Mathematical Foundation of Game Design and Animation

Lecture 1

Coordinate spaces

Dr. Paolo Mengoni
pmengoni@hkbu.edu.hk
Senior Lecturer @HKBU Department of Interactive Media

Agenda

- Why multiple coordinate spaces?
- Some useful coordinate spaces.
- Coordinate space transformations.
- Camera Space

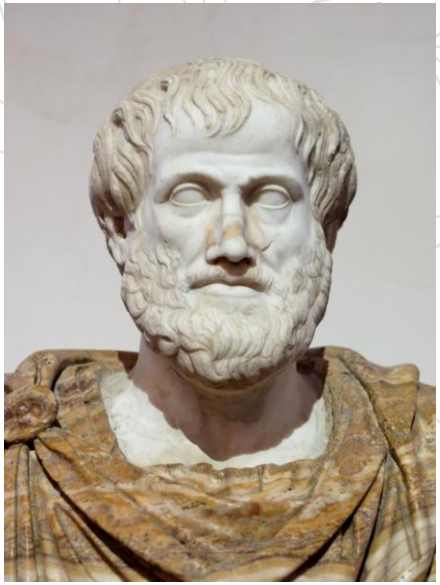
Why Multiple Coordinate Spaces?

Why Multiple Coordinate Spaces?

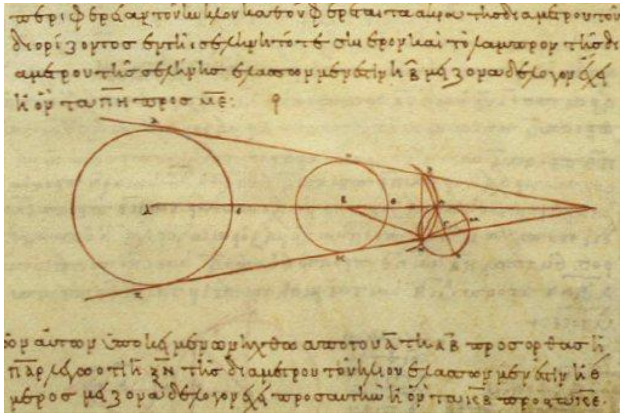
- Some things become easier in the correct coordinate space.
- There is some historical precedent for this observation (next slide).
- We can leave the details of transforming between coordinate spaces to the graphics hardware.

Astronomic coordinates

- Aristotle (384-322 BCE) proposed a geocentric universe with the Earth at the origin.



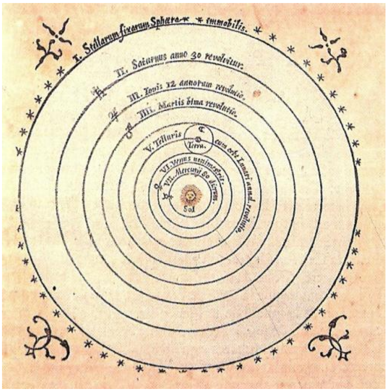
- Aristarchus (ca. 310-230 BCE) proposed a heliocentric universe with the Sun at the origin.



(Images from Wikimedia Commons)

Astronomic coordinates

- Nicholas Copernicus (1473-1543) observed that the orbits of the planets can be explained more simply in a heliocentric universe.



Some Useful Coordinate Spaces

Some Useful Coordinate Systems

- In a 3D game we may find useful the following coordinate spaces

- World space
- Object space
- Camera space
- Upright space

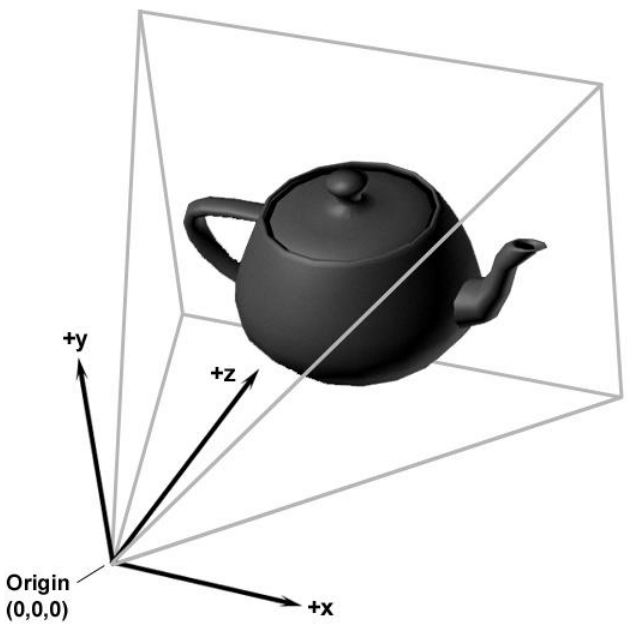
World Space

- World space is the global coordinate system.
- Use it to keep track of position and orientation of every object in the game.
  - One space for all the objects.

Object Space

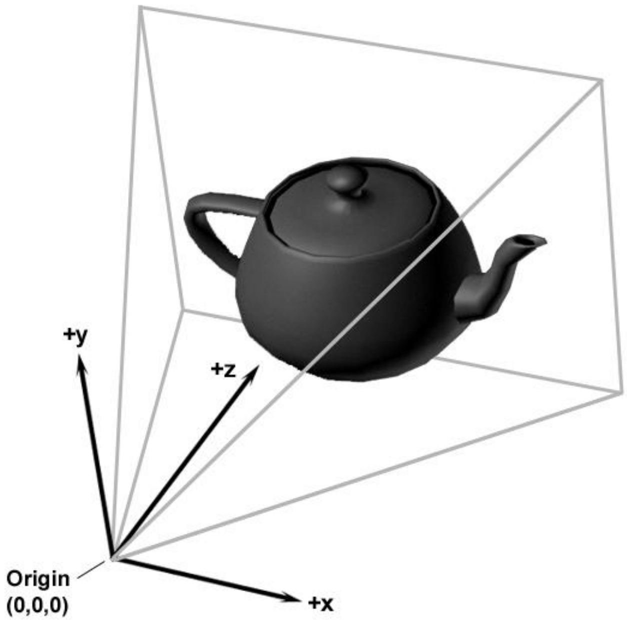
- In the Object Space every object in the game has its own coordinate space with:
  - Its own origin (where it is),
  - Its own concept of “up” and “right” and “forwards”,
- Use it to keep track of relative positions and orientation
  - Useful for collision detection, AI, etc.

Camera Space



- Object space for the viewer, represented by a camera, used to project 3D space onto screen space

Camera Space Terminology



- Frustum: the pyramid of space seen by the camera
- Clipping: objects partially on screen
- Occlusion: objects hidden behind another object
- Visibility: inside or outside of frustum, occluded, clipped

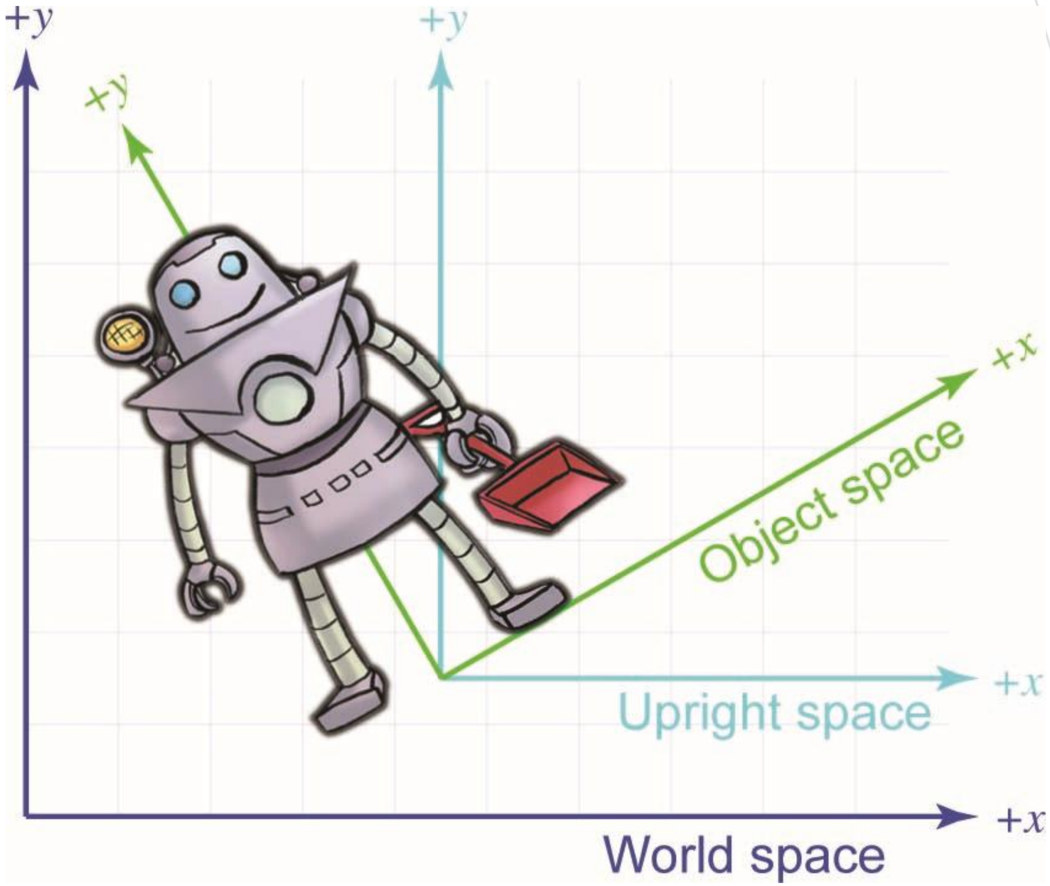
Upright Space

- Upright space is in a sense “halfway” between world space and object space.

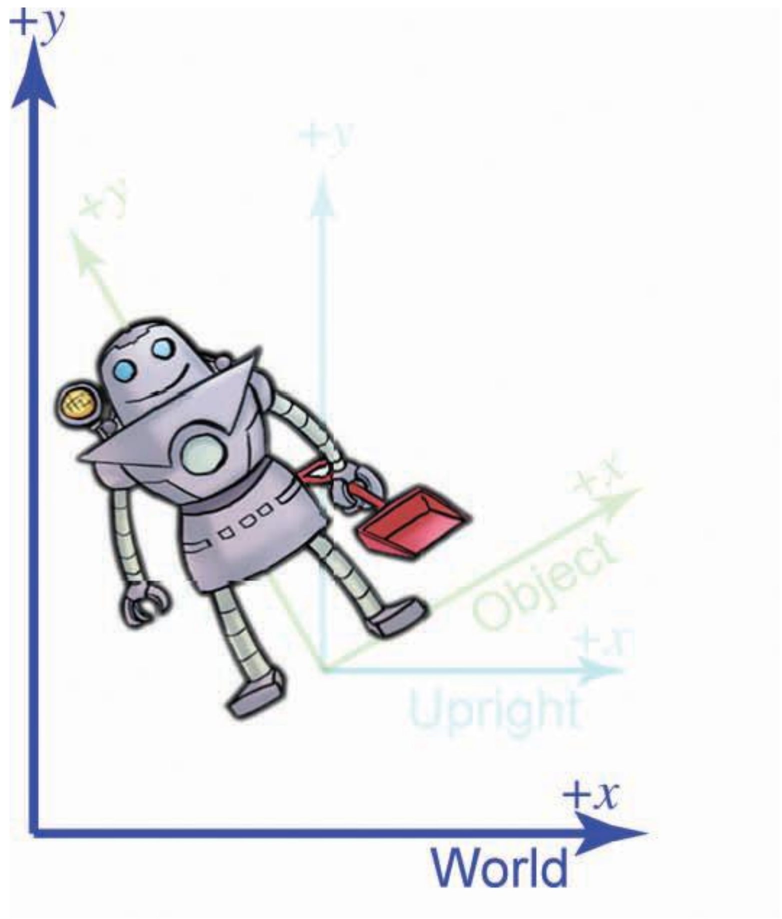
- Upright space has
  - Object space origin
  - World space axes

- No standard for it, but useful in certain situations.

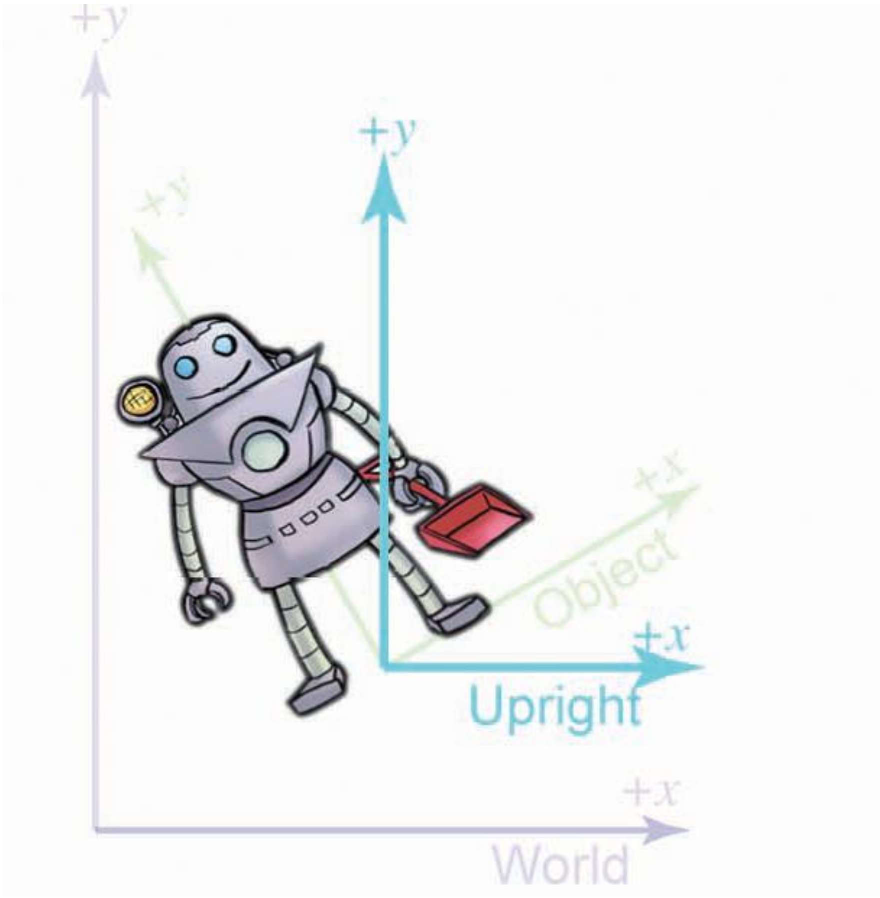
Object, Upright, and World Space



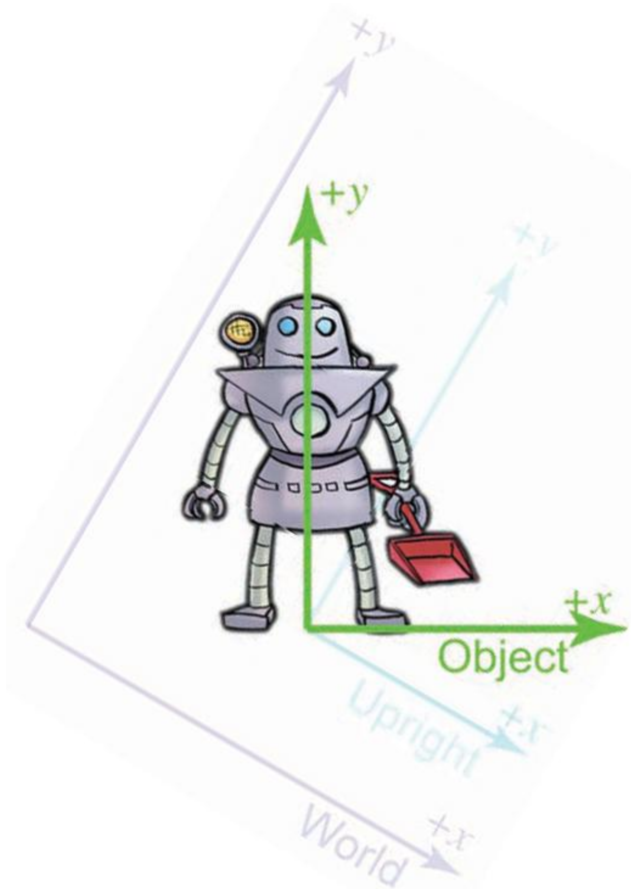
Robot in World Space



Robot's Upright Space



Robot’s Object Space



Why Upright Space?

- It separates translation and rotation
  - It is a handy visualization tool
  - It is inspired by both math and hardware implementation

- Translate between world and upright space.

- Rotate between upright and object space.

- Which brings us to *coordinate space transformations*…

Coordinate Space Transformations

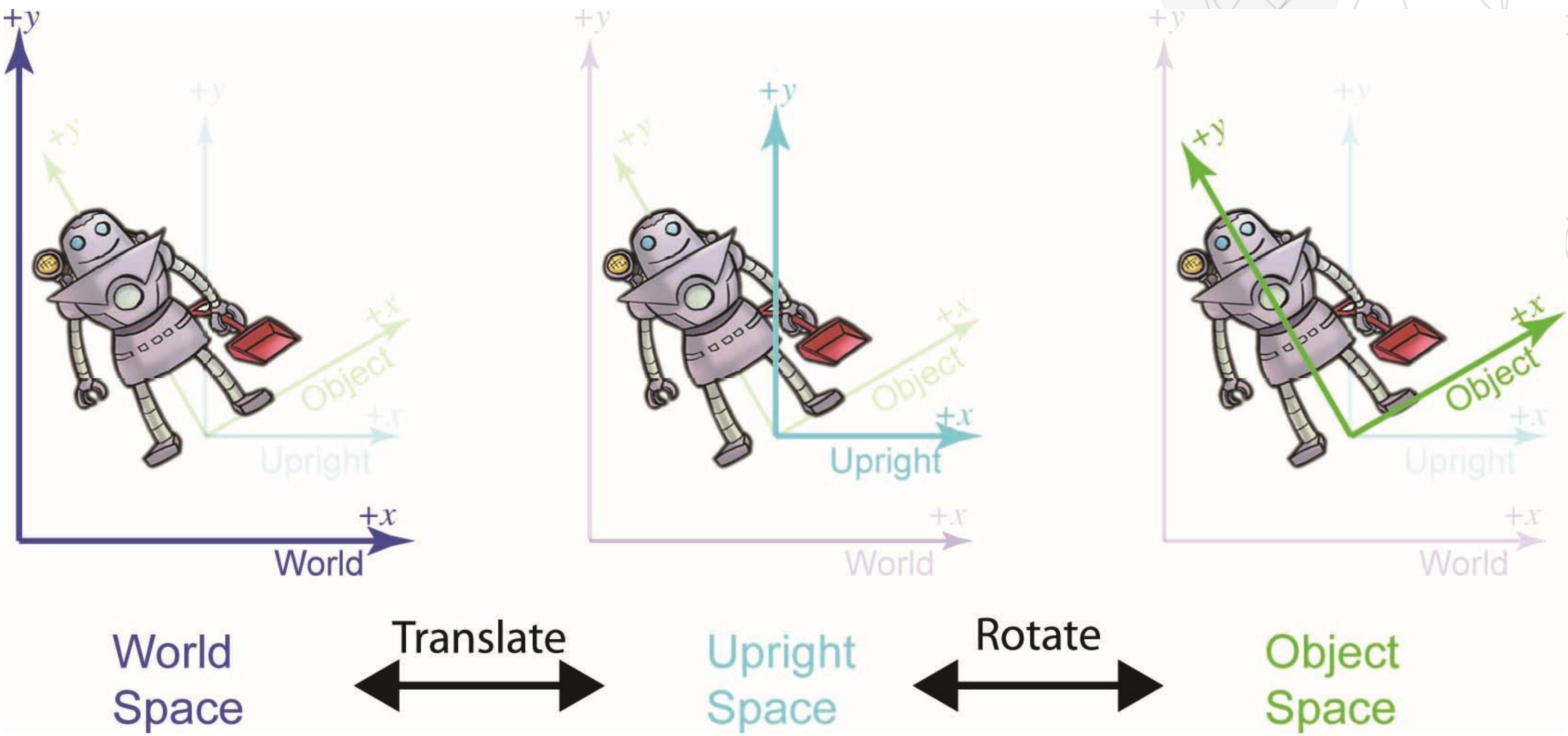
Coordinate Space Transformation

- Two important types of transformation used to transform one coordinate space into another:

- Translation
  - Changes position in space
  - Gives new location of origin

- Rotation
  - Changes orientation in space
  - Gives new directions of axes

- Which one comes first?
  - Do rotation first if the object is at the origin

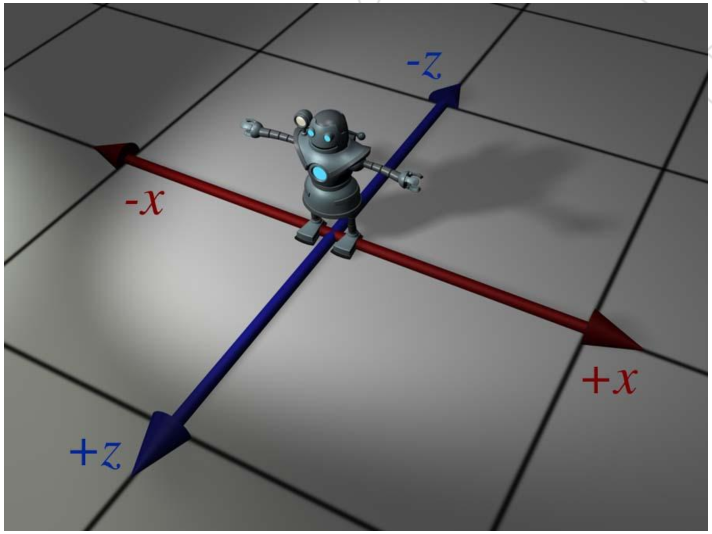

+y
+x
World
Upright
Object
World Space
Translate
Upright Space
Rotate
Object Space

Example

- Let's say that we are working for an advertising agency that has just landed a big account with a food manufacturer.
- You are assigned to the project to make a slick computer-generated ad promoting one of their most popular items, *Herring Packets*, which are microwaveable herring ~~sandwiches~~ food products for robots.
- Of course, the client has a tendency to want changes made at the last minute, so we need to be able to get a model of the product at any possible position and orientation.

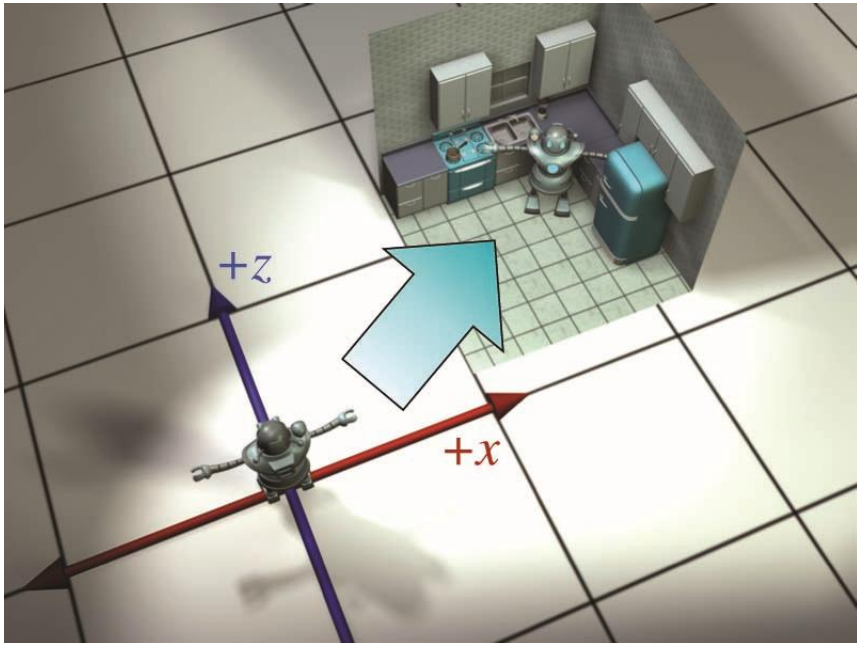
Start with the Artist's Model

- For now, because we have the model in its home position, object space and world space (and upright space) are all the same by definition.
- For all practical purposes, in the scene that the artist built containing only the model of the robot, world space is object space.



Goal Transformation

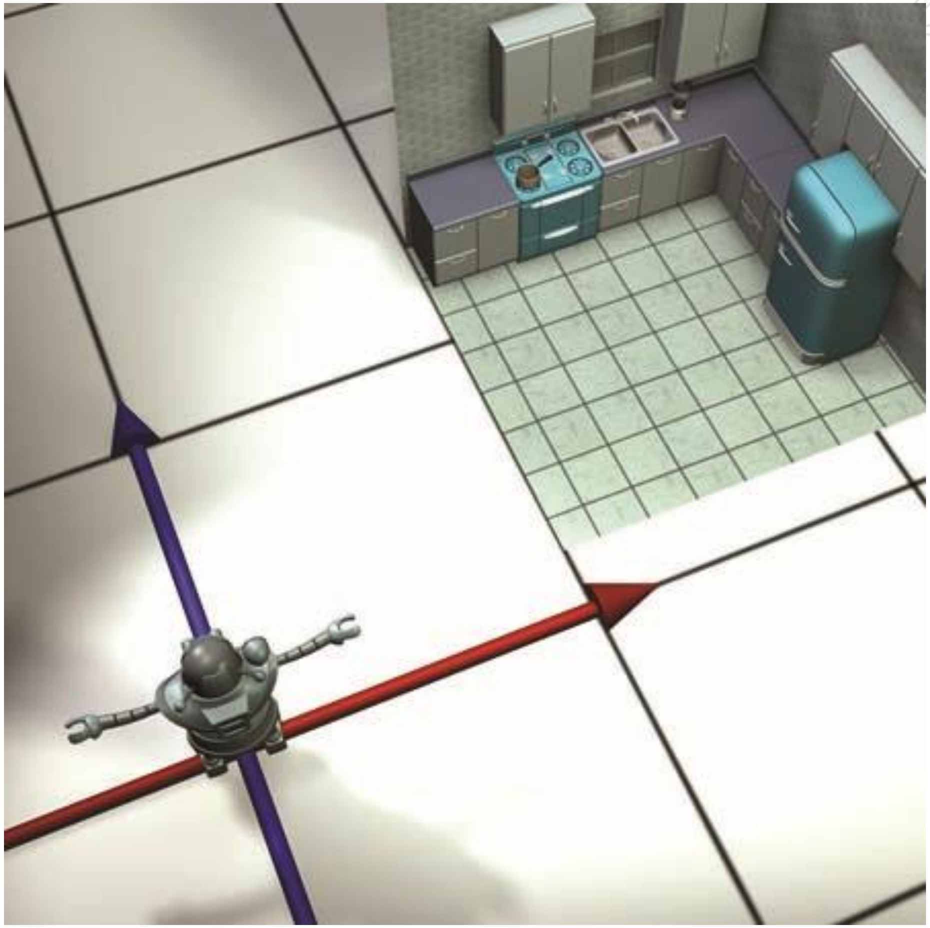
- Our goal is to transform the vertices of the model from their home location to some new location (in our case, into a make-believe kitchen), according to the desired orientation (1st in this case) and position (2nd in this case) of the robot based on the executive's whims at that moment.



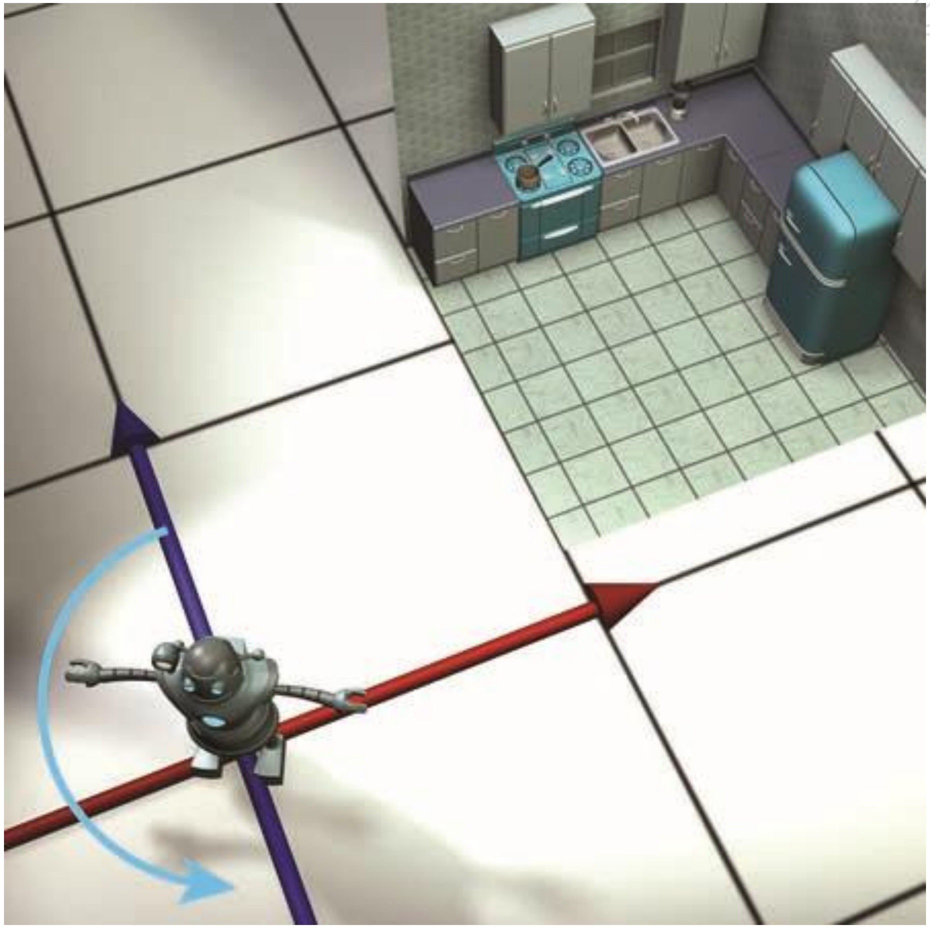
More Details

- Let's talk a bit about how to accomplish this.
- Conceptually, to move the robot into position we first rotate her clockwise 120°
  - i.e., by “heading left 120°
- Then we translate 18ft east and 10ft north, which according to our conventions is a 3D displacement of [18, 0, 10].

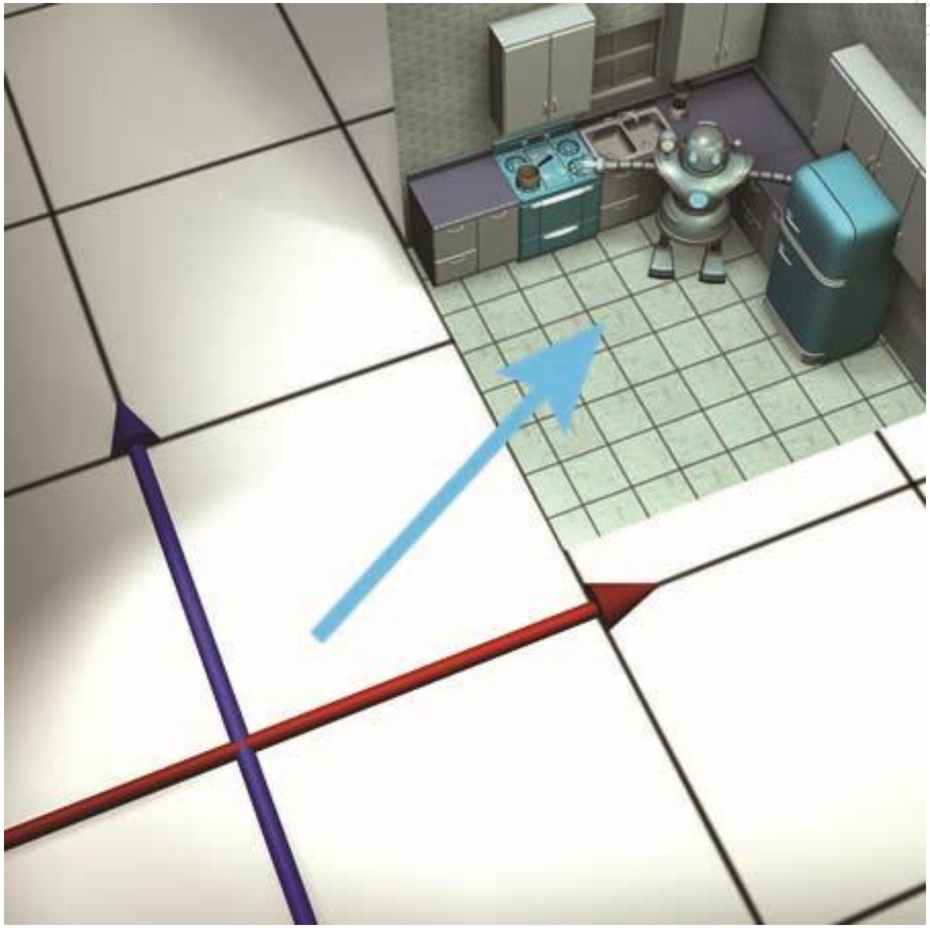
Original Position



Rotate



Then Translate



Rotate Before Translate

- Why rotate before we translate?

- Rotation about the origin is easier – translating first would mean that we have to rotate around an arbitrary point.

- Rotation about the origin is a linear transform.
  - Later more on this
- Rotation about an arbitrary point is an affine transform.
  - Much later more on this

- Affine transforms can be expressed as a sequence of primitive operations.

Camera space

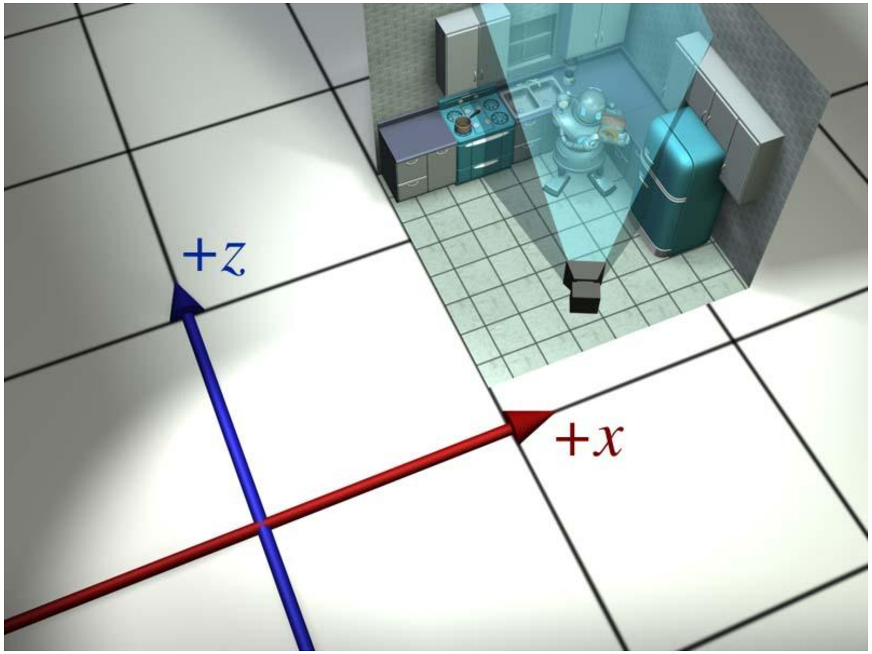
Camera Space

- So we've managed to get the robot model into the right place in the world.

- But to render it, we need to transform the vertices of the model into *camera space*.

- In other words, we need to express the vertices' coordinates relative to the camera.
  - E.g., if a vertex is 9ft in front of the camera and 3ft to the right, then the z and x coordinates of that vertex in camera space would be 9 and 3, respectively.

Where the camera is



What the camera sees

Camera Space

- It’s easier to reason about a camera at the origin, looking along a primary axis.

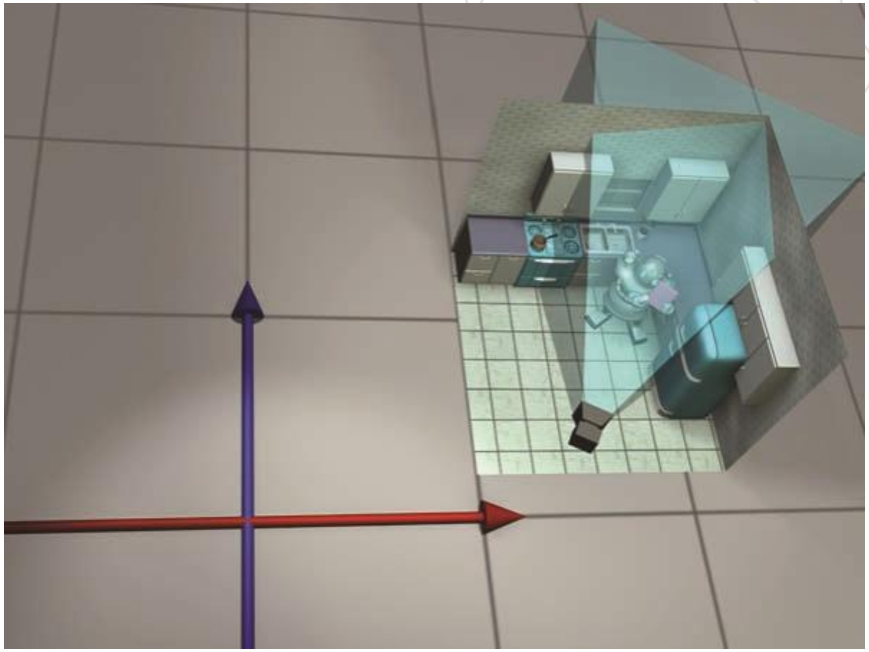
- So, we move the whole world so that the camera is at the origin.

- First, we translate. Then, we rotate.
  - For the same reason as before, because rotation about the origin is easier than rotation about an arbitrary point.

Original Position

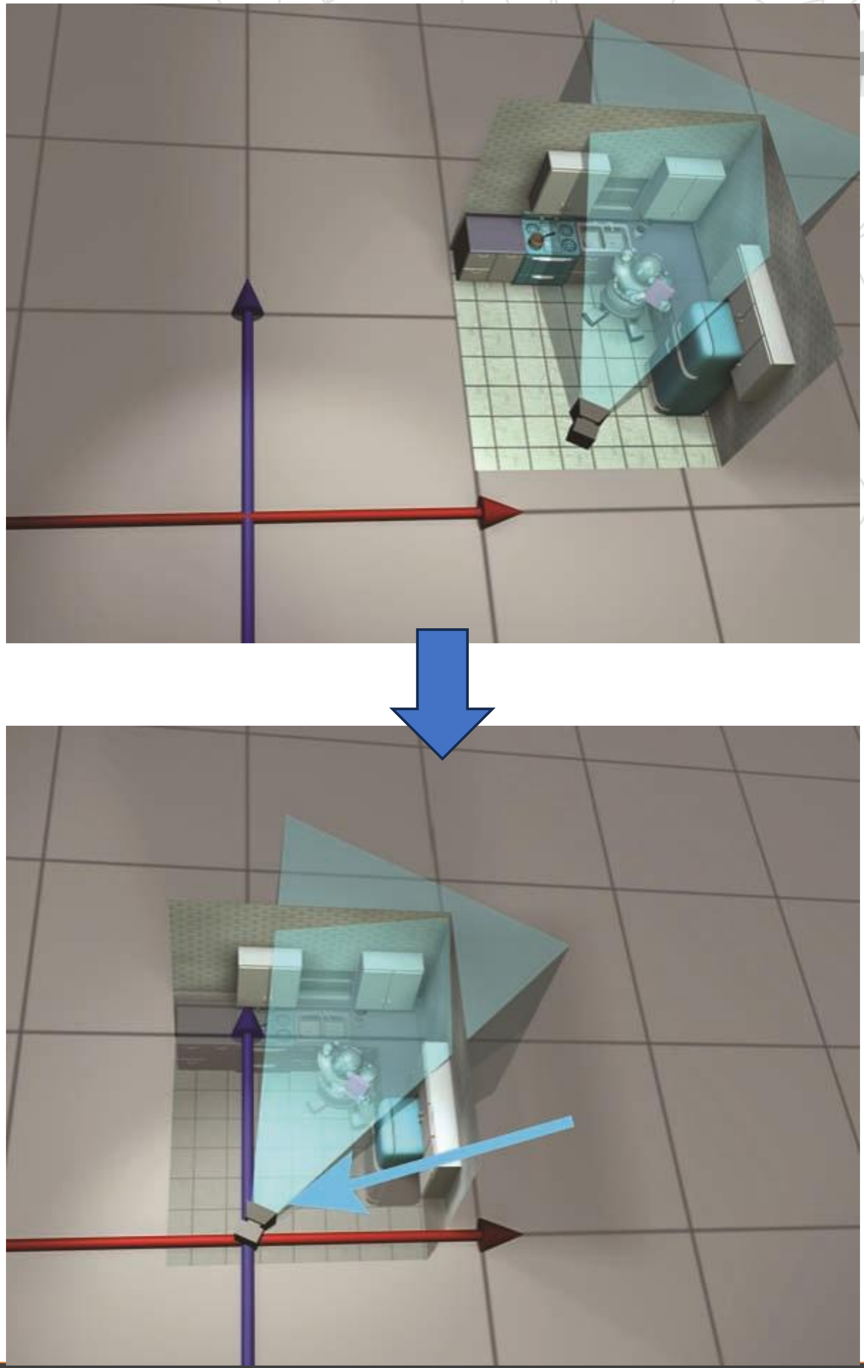
- Transform to Camera Space

- We use the opposite translation and rotation amounts, compared to the camera's position and orientation.
- In the Figure, the camera is at (13.5, 4, 2).



Translate the World

- Transform to Camera Space
- In the Figure, the camera is at (13.5, 4, 2).
- So, to move the camera to the origin, we will translate by [-13.5, -4, -2].

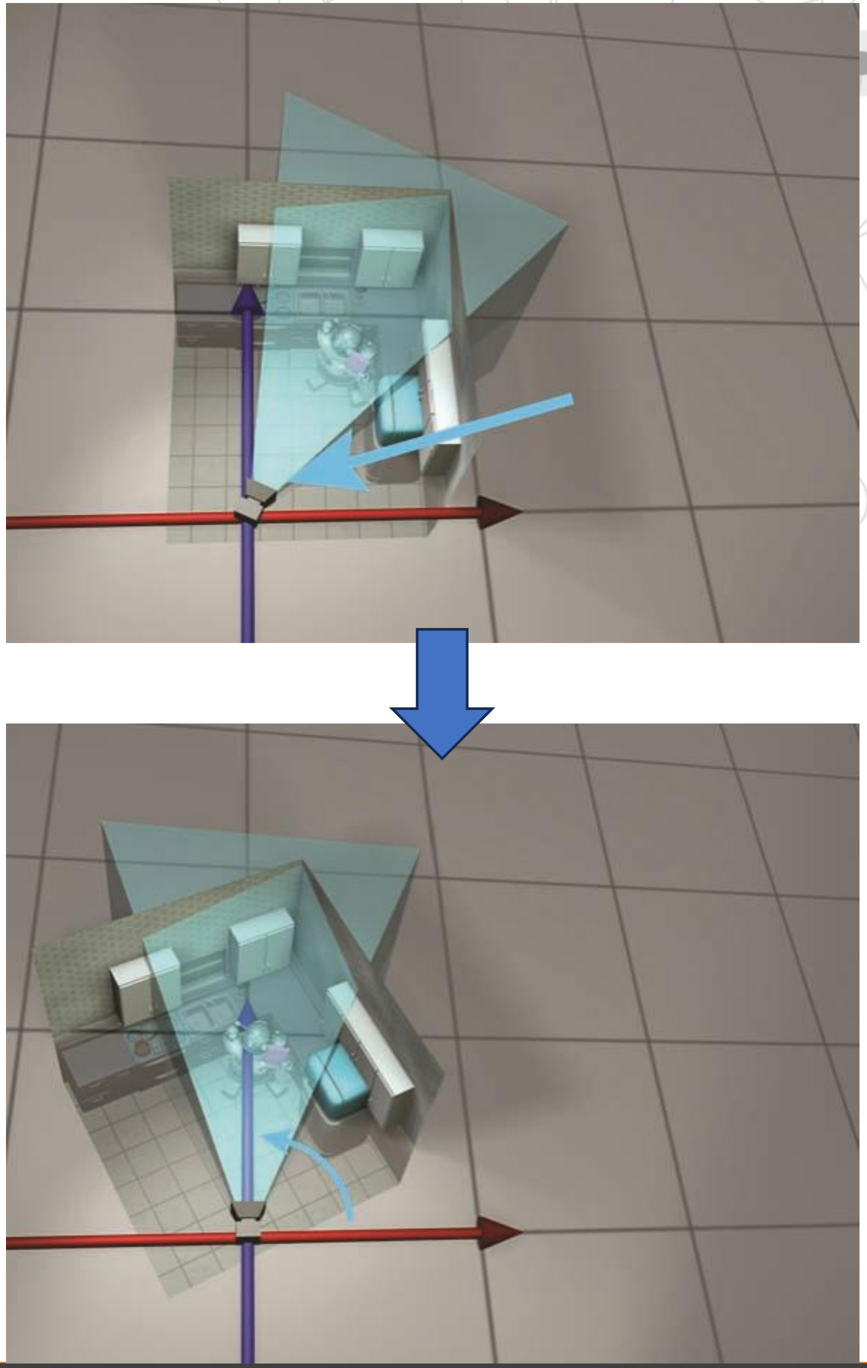


Rotate the World

- Transform to Camera Space

- The camera is facing roughly northeast and thus has a clockwise heading compared to north.

- A counter-clockwise rotation is required to align camera space axes with the world space axes.



Notes

- The *world space* → *camera space* transform is usually done inside the rendering system, often on a dedicated graphics processor.
- Camera space isn't the “finish line" as far as the graphics pipeline is concerned. From camera space, vertices are transformed into clip space and finally projected to screen space.

Camera Space perspective

- The camera consists of a near plane that represents the screen where pixels are drawn, and a far plane that determines how far back into the world the camera can see.

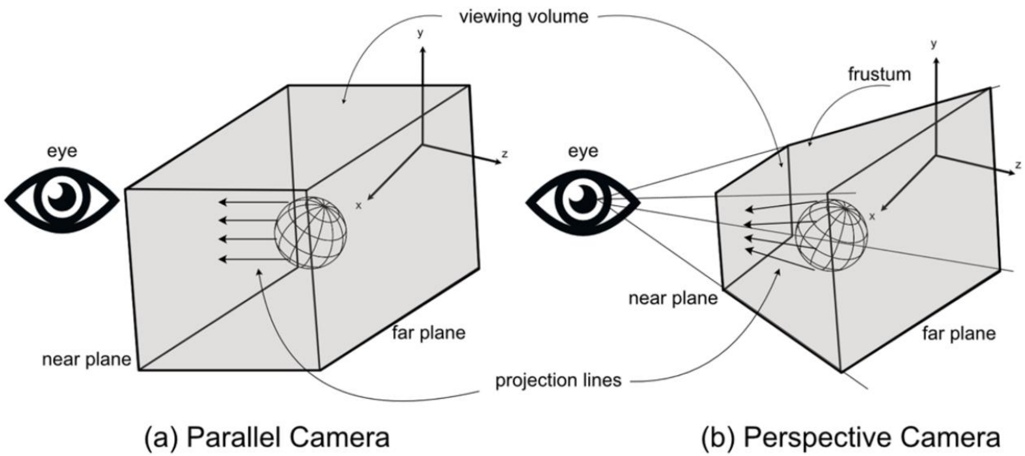


Figure 4.7: A camera with a perspective view

Camera Space perspective

- The area contained between the near and far planes is called the viewing volume.

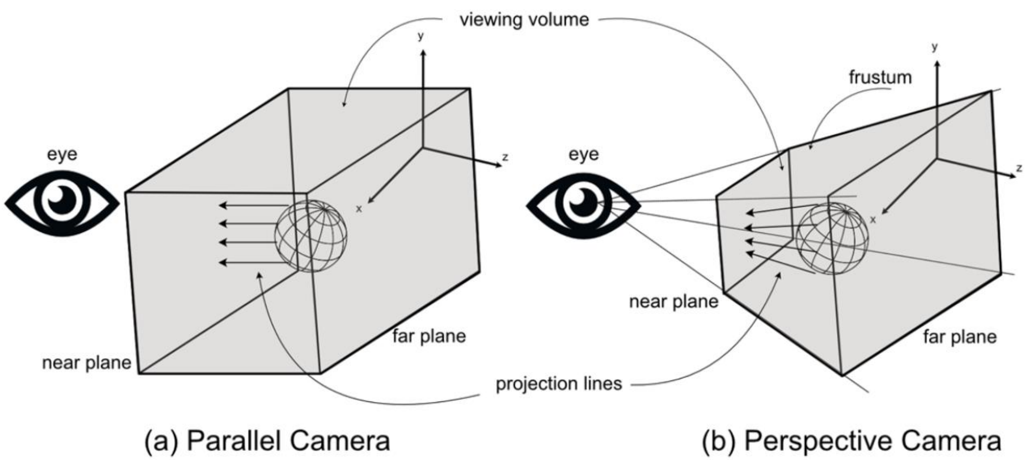


Figure 4.7: A camera with a perspective view

- The object outside the viewing volume are culled.

Camera Space perspective

- The area contained between the near and far planes is called the viewing volume.

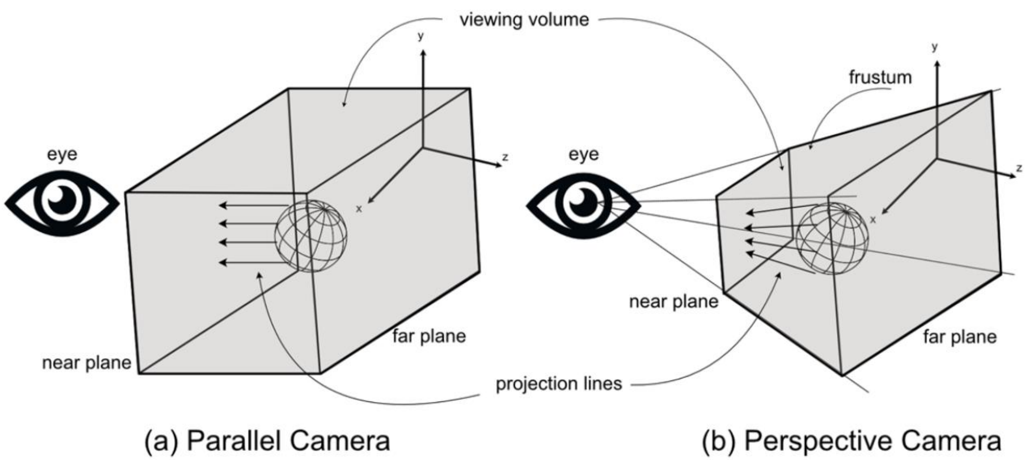


Figure 4.7: A camera with a perspective view

- The object outside the viewing volume are culled.

Camera Space perspective

- The shape of the viewing volume differs between camera types and influences how a scene is rendered on the screen.

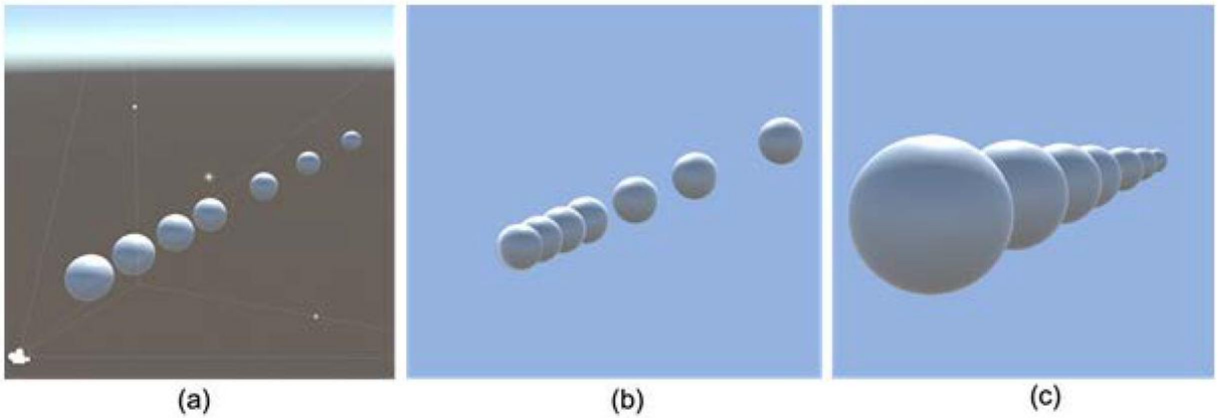


Figure 4.8: A 3D scene (a) viewed in parallel (b) and perspective (c)

- Essentially, there are two camera types:
  - parallel (i.e., orthogonal)
  - perspective.

Camera Space perspective

- The shape of the viewing volume is a:
  - Rectangular prism for a parallel camera
  - Frustum (i.e., a four-sided pyramid with the top cut off) for perspective camera.

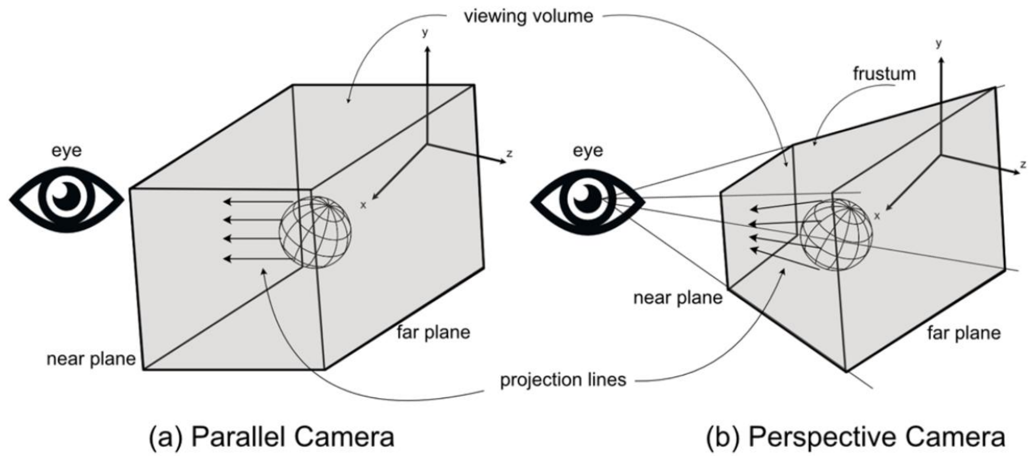


Figure 4.7: A camera with a perspective view