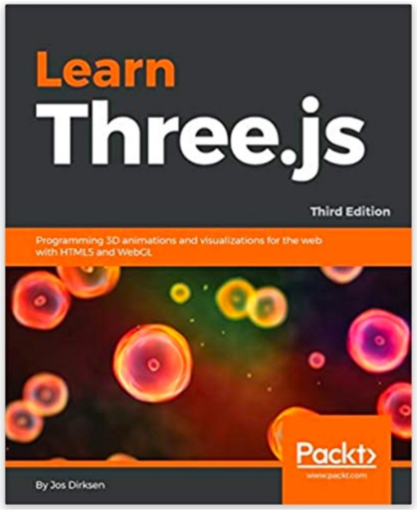
COMPUTER GRAPHICS


Learn Three.js

Third Edition

Programming 3D animations and visualizations for the web with HTML5 and WebGL

Packt>

www.packt.com

By Jos Dirksen

* Adapted from CISC 3326 lecture by Michael Mandel

CAMERAS

Based on this CS 307 reading and this CS 307 lecture*

EXERCISES – THREE.JS

Exercise: Perspective Projection

- Assume a synthetic camera with the image plane (near plane) at a distance d from the center of projection (COP).
- Suppose the scene contains a tree of height H at a distance D from the COP.
- What is the height of the projected tree?

Exercise: Perspective Projection

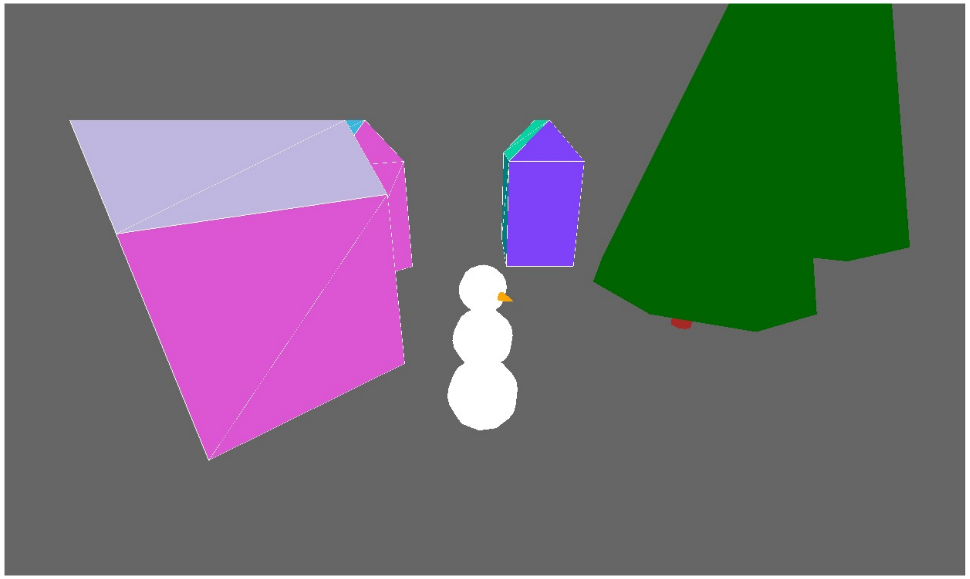
- Camera helper demo

Exercises: Setting up a Camera

- Using this town-view-before
- Set up a camera to view the snowman from above and to the right.
- You might find it helpful to use the following town-view-gui.
- Define a function to set up the camera the way you want.
- Replace TW.cameraSetup() with your new camera.

Exercises: Setting up a Camera

- Target view:

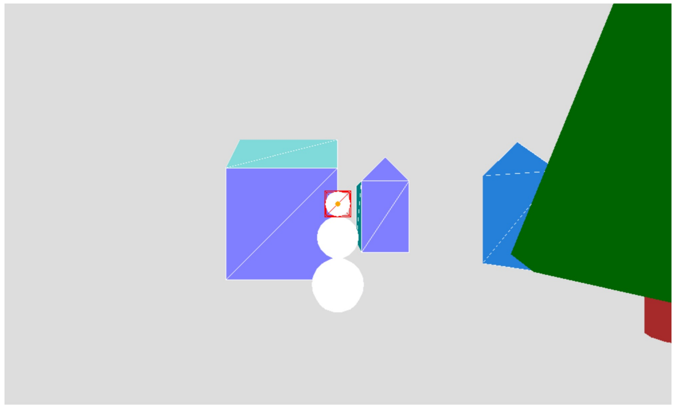


Exercise: Zooming vs. Dollying

- Is there any difference between
  - *zooming in,* keeping the eye point the same and reducing the FOV
  - *dollying in,* keeping the FOV the same and moving the camera closer
- Let's try to experience the effects of these changes to the camera setup.

Zooming vs Dollying

- Add a wireframe box to the scene that fits snuggly around the head of the snowman.
- Then try to set the camera parameters so that the scene appears like this:
- Initial snowman view:

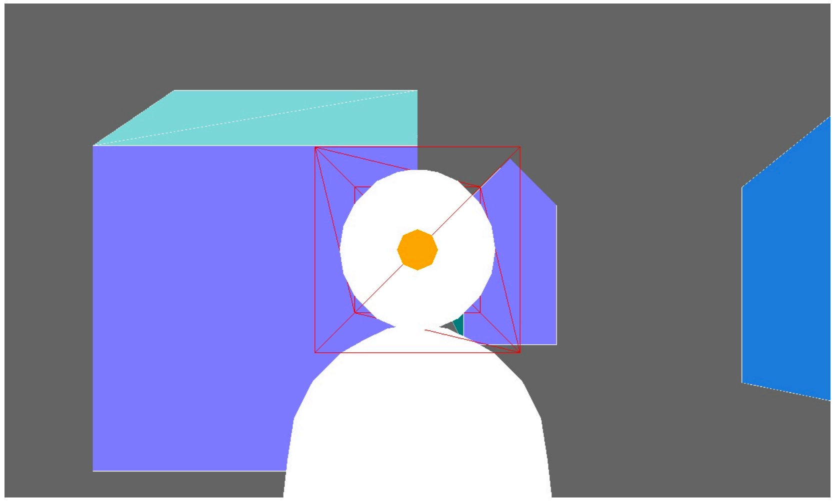


Zooming vs Dollying hints

- **Hints:** the camera is positioned 5 units in front of the center of the snowman’s head
  - at the same x and y coordinates as this center point
- and is *looking at* this center point.
- Set the FOV to achieve the above appearance.

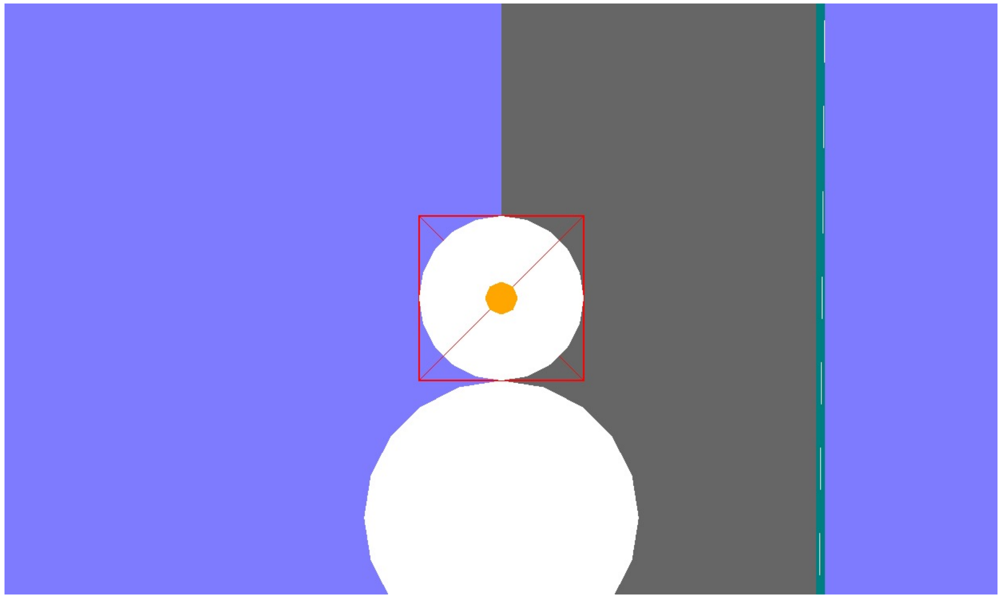
Zooming vs Dollying: Setting 1

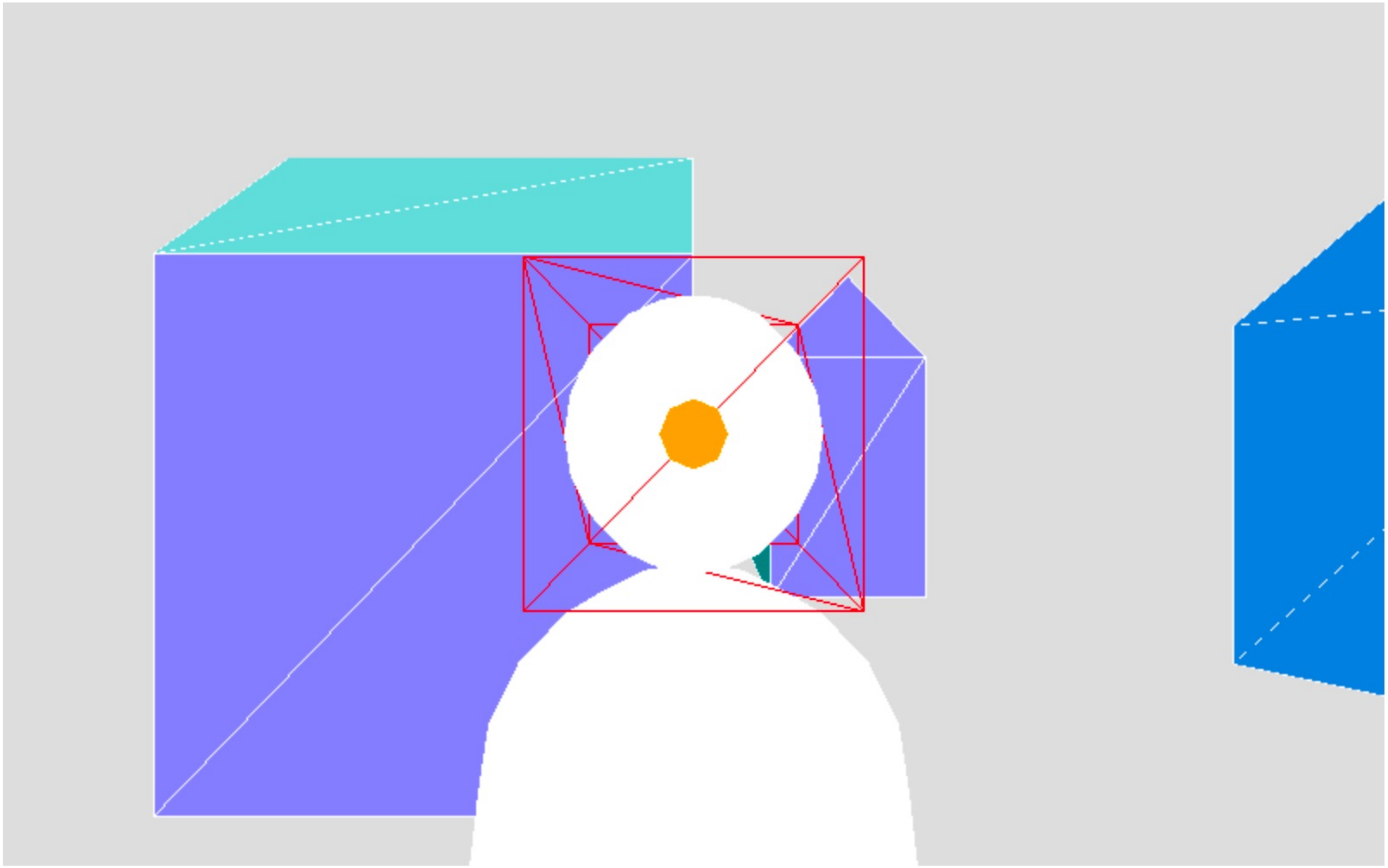
- Try to set the camera parameters to reproduce each of the following views
- One requires dollying, the other zooming
- Target view 1:



Zooming vs Dollying: Setting 2

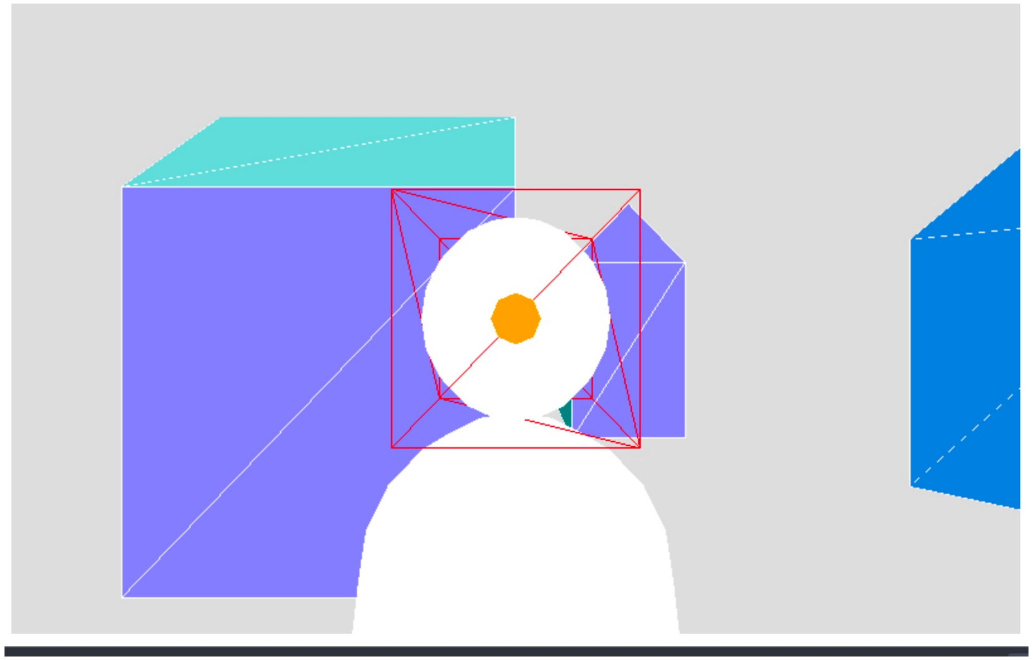
- Try to set the camera parameters to reproduce each of the following views
- One requires dollying, the other zooming
- Target view 2:



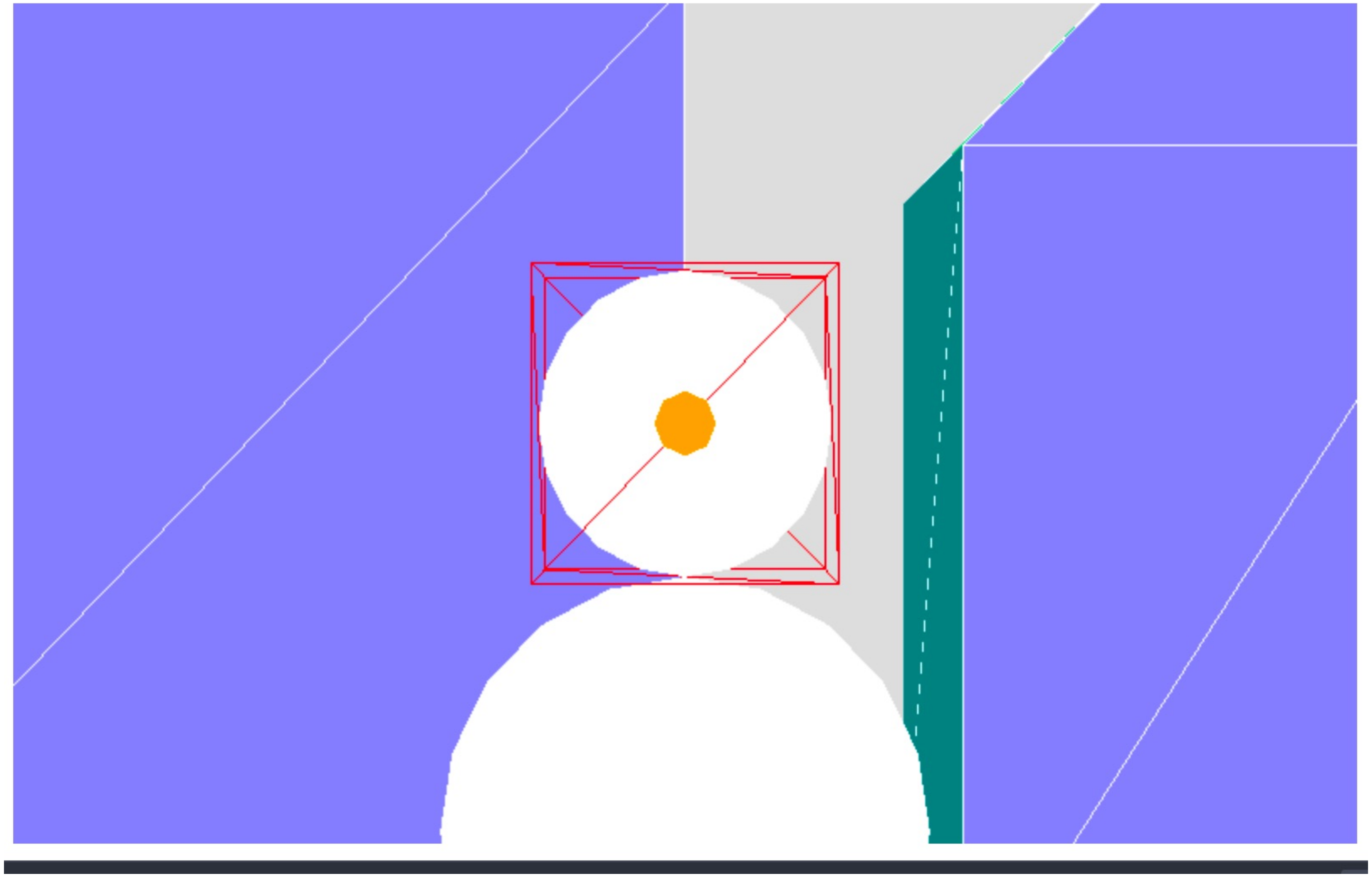


Zooming vs Dollying solution

- Your dolyying solutions might be like this:



Zooming solutions:



Vup (Vector View-Up)

- Most of the time, Vup is very simple: we have
  - *V*up=(0,1,0)
- If we want something different, though, Vup can be confusing.

Vup

- Try to imagine it this way:
  - Visualize the Vup vector in 3D space, along with the image plane.
  - *Project* the Vup vector onto the image plane
  - The frustum spins around the VPN (view plane normal)
    - so that the Vup vector is *parallel to the left/right edges*.
  - When the top of the frustum is mapped onto the canvas, the Vup vector is parallel to the left/right edge of the canvas.

Vup

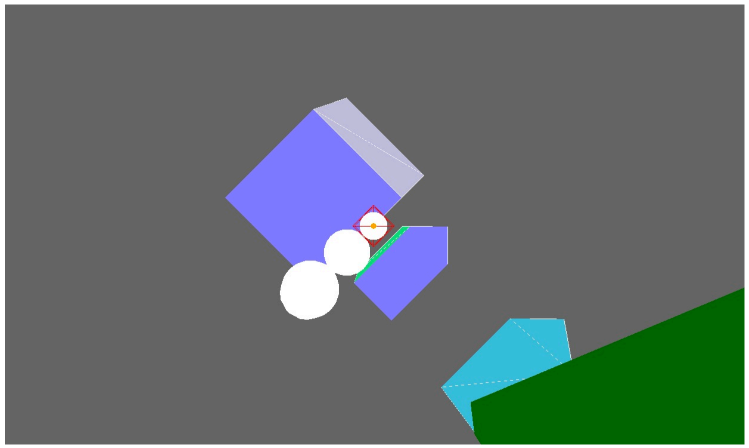
- There are two important consequences of the way Vup works:
  - The Vup vector can’t project to a point

    => it can’t be parallel or anti-parallel to the VPN.
  - If Vup vector change doesn’t change the direction of its projection on the image plane
    - =>*doesn’t* affect anything.
      - The image is still oriented the same way.

Exercise: Changing Vup

- Let’s try to experiment with changing Vup
- Try to reproduce the below view
  - Think about what Vup might be.
  - Set up the camera for each scene.

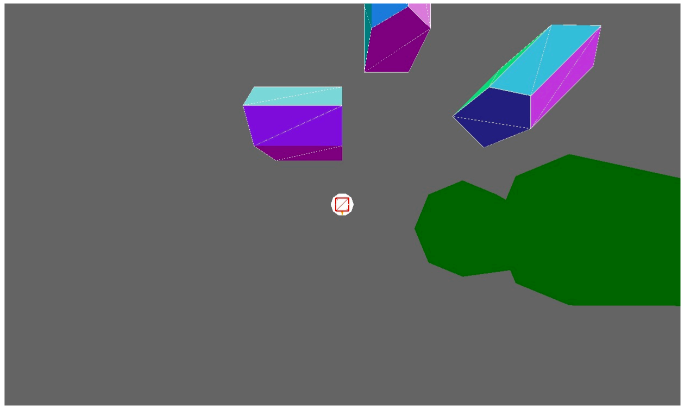
- Taget view 1:



Exercise: Changing Vup 2

- Let’s try to experiment with changing Vup Try to reproduce the below view Think about what Vup might be.
- Set up the camera for each scene.

- Target view 2:



Questions?