Introduction to the Java3D View Model

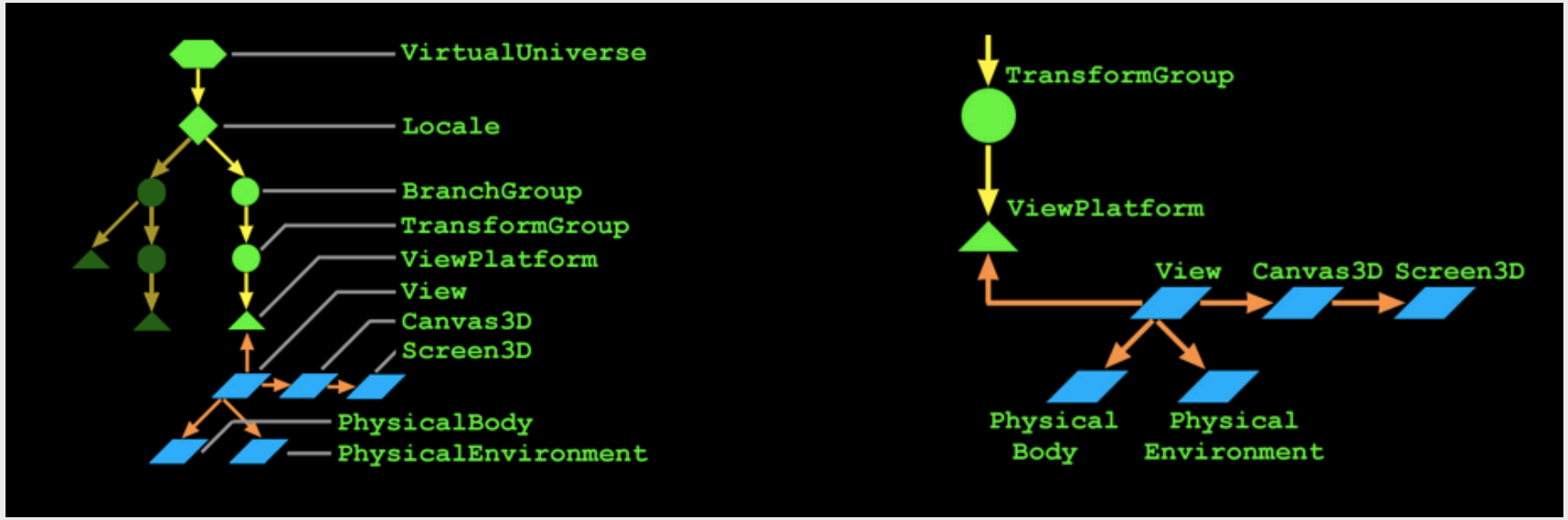
ConfiguredUniverse

Motivation

- The Java3D view model contains descriptions of the physical and the virtual world
- We need control over the user's virtual position and orientation
  - Navigate their viewpoint using the mouse, or any other input device
  - Or move the viewpoint automatically in a guided tour
  - We call such a user viewpoint a view platform
- We also need a careful abstraction from hardware gadgetry
  - Support different display configurations
    - Stereo, HMDs, multi-screen portals
  - Support head tracking
- You will also understand the difference between the Java3D view model and OpenGL

view branch

- Viewing controls are typically placed in a parallel view branch of the scene graph



Coexisting in the physical and virtual worlds

- Shapes, branch groups, locales, and the virtual universe define the virtual world
- A user co-exists in this virtual world and in the physical world
  - The user has a position and orientation in the virtual world
  - The user, and their display, have positions and orientations in the physical world
- The Java 3D view model handles mapping between virtual and physical worlds

Understanding constraints and policies

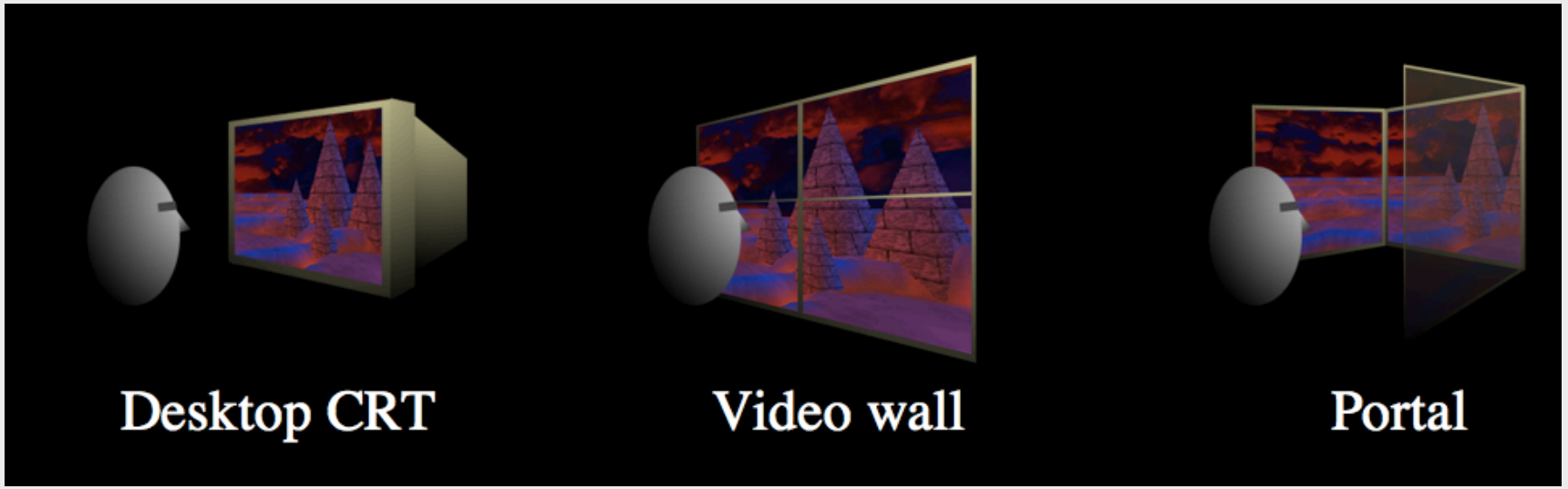
- A chain of relationships control this mapping between worlds
    - Eye locations relative to the user's head
    - Head location relative to a head tracker
    - Head tracker relative to the tracker base
    - Tracker base relative to an image plate (display)
    - *. . . and so on, with variations*
- A *constraint system* defines these relationships
    - For a given environment and usage, some relationships are constants, while others vary

Understanding view policies

- The *view policy* selects one of two constraint systems
  - *Room-mounted displays*
    - Displays whose locations are fixed
    - CRTs, video projectors, multi-screen walls, portals
  - *Head-mounted displays*
    - Displays whose locations change as the user moves
    - Head Mounted Displays (HMDs)

Understanding room-mounted displays

- In a room-mounted display, the user looks at a display with a fixed location relative to the physical world

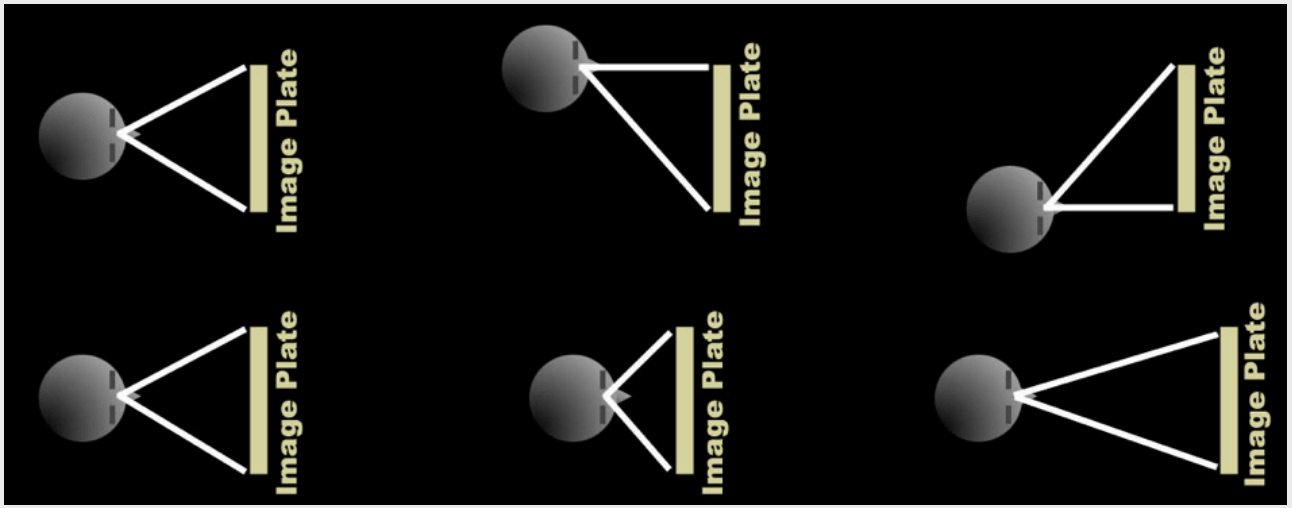


Understanding room-mounted displays

- Physical world components include:
  - *Head* - the user!
  - *Eye* - a "center eye" on the user's head
  - *Image plate* - the physical display
  - *Head tracker* - the tracked point on a user's head
  - *Tracker base* - the tracking system's emitter or reference point

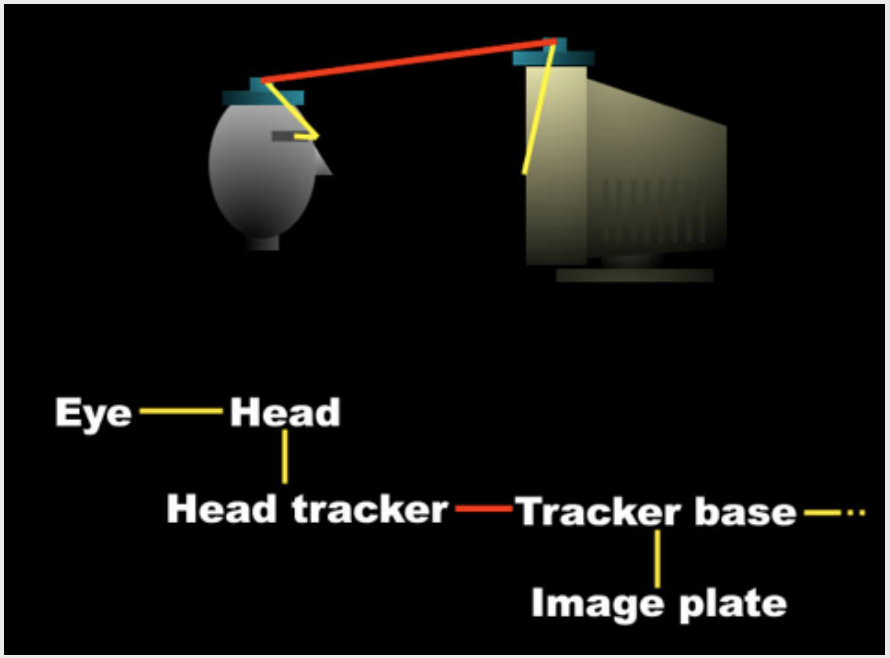
Understanding room-mounted displays

- The constraint system uses the eye location relative to the image plate to compute a correct view frustum
    - When using head tracking, eyepoint is computed automatically
    - When not using head tracking, eyepoint may be set manually



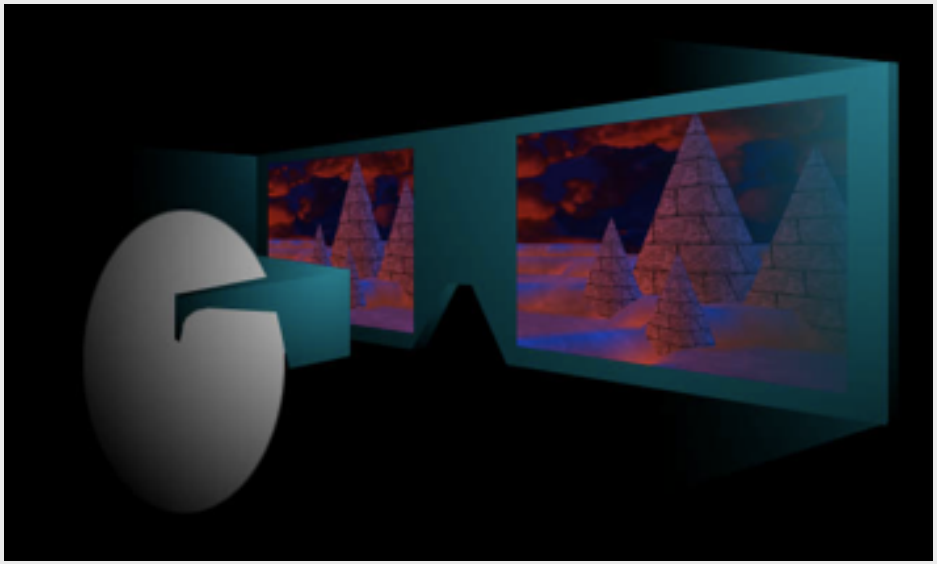
Understanding room-mounted displays

- To map from eye to image plate, the constraint system uses a chain of coordinate system mappings



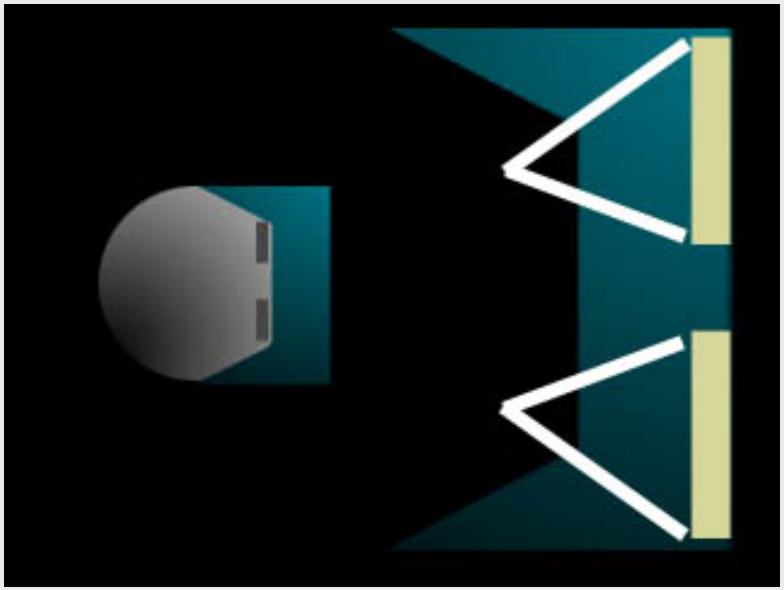
Understanding head-mounted displays

- In a *head-mounted display*, each eye looks at its own display with a fixed location relative to the user's head



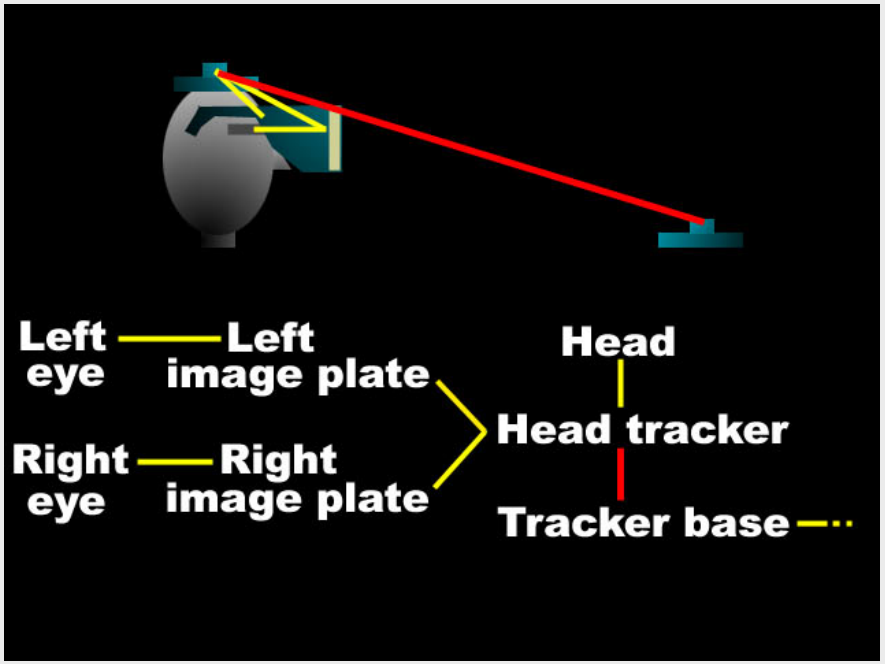
Understanding head-mounted displays

- The constraint system uses the left and right eye locations relative to the left and right image plates to compute correct view frustums



Understanding head-mounted displays

- To map from left and right eyes to their image plates, the constraint system uses a chain of coordinate system mappings

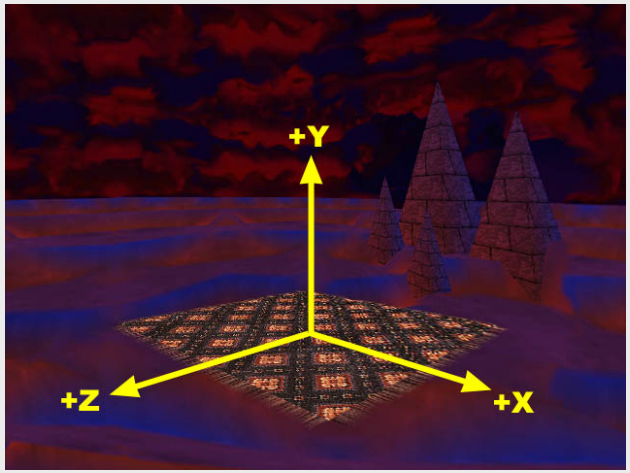


Understanding physical to virtual mappings

- Recall that the user co-exists in the virtual and physical worlds
  - The user has a physical position and orientation
  - The user also has a virtual position and orientation
- Room- and head-mounted display view policies handle mapping from the user's physical body to a tracker base and image plates
- To map from this physical world to the virtual world, we add to the constraint chain:
  - Tracker base to coexistance
  - Coexistance to view platform
  - View platform to locale
  - Locale to virtual universe

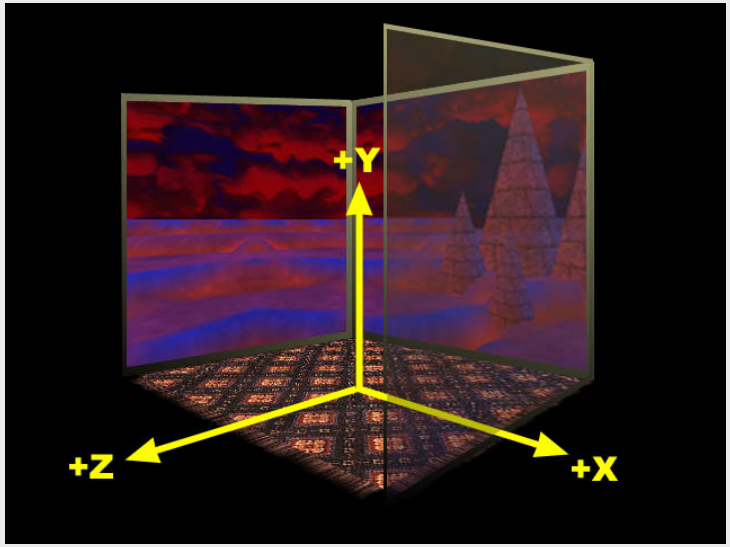
Understanding physical to virtual mappings

- For example, in a virtual world imagine the view platform is a magic carpet
  - The user can walk about on the carpet
  - The carpet flies about under application control
  - Define the view platform origin at "ground level", at carpet center



Understanding physical to virtual mappings

- In the physical world, imagine the user is standing in a portal
  - Images of the virtual world are rendered on three sides
  - The user's position is tracked within the portal
  - Define the portal origin at ground level, at the portal center



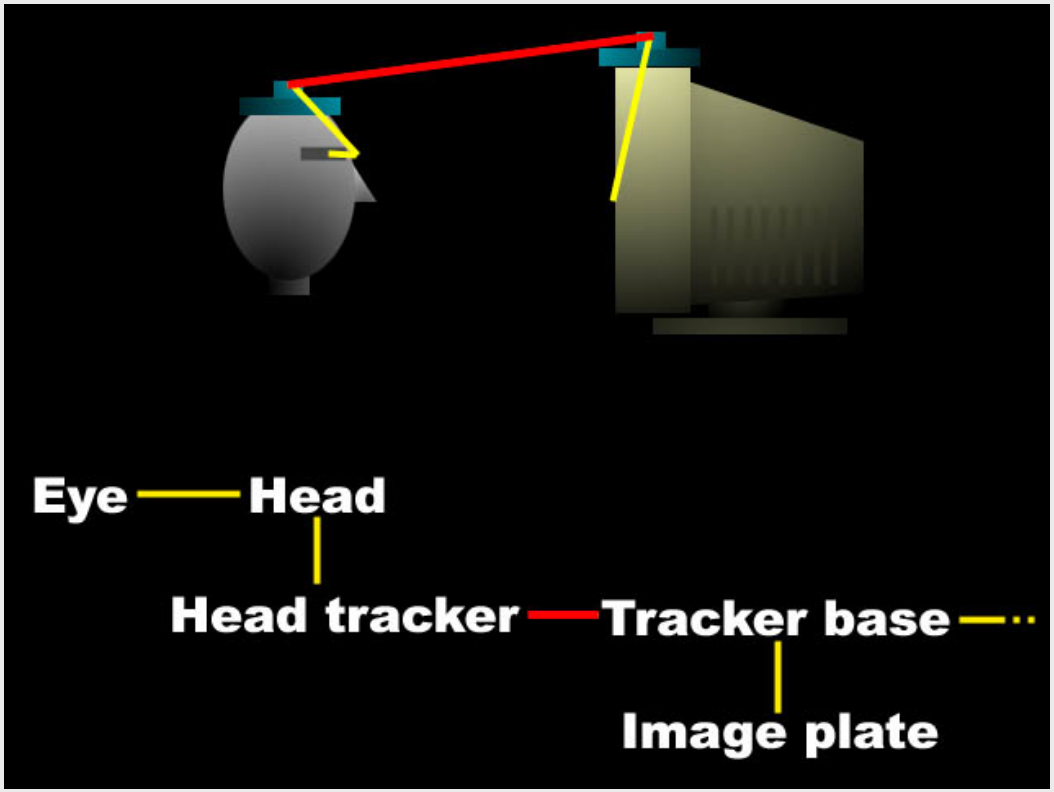
Understanding physical to virtual mappings

- Physical device configurations and a room-mounted view policy establish:
  - Mappings from eye to head, to head tracker, to tracker base, to image plate (portal screen)
  - A tracker base to coexistence transform maps from the tracker base to the portal center
    - Or whatever reference point you prefer
- As the user moves about, their location is computable relative to this coexistence frame of reference - the portal center

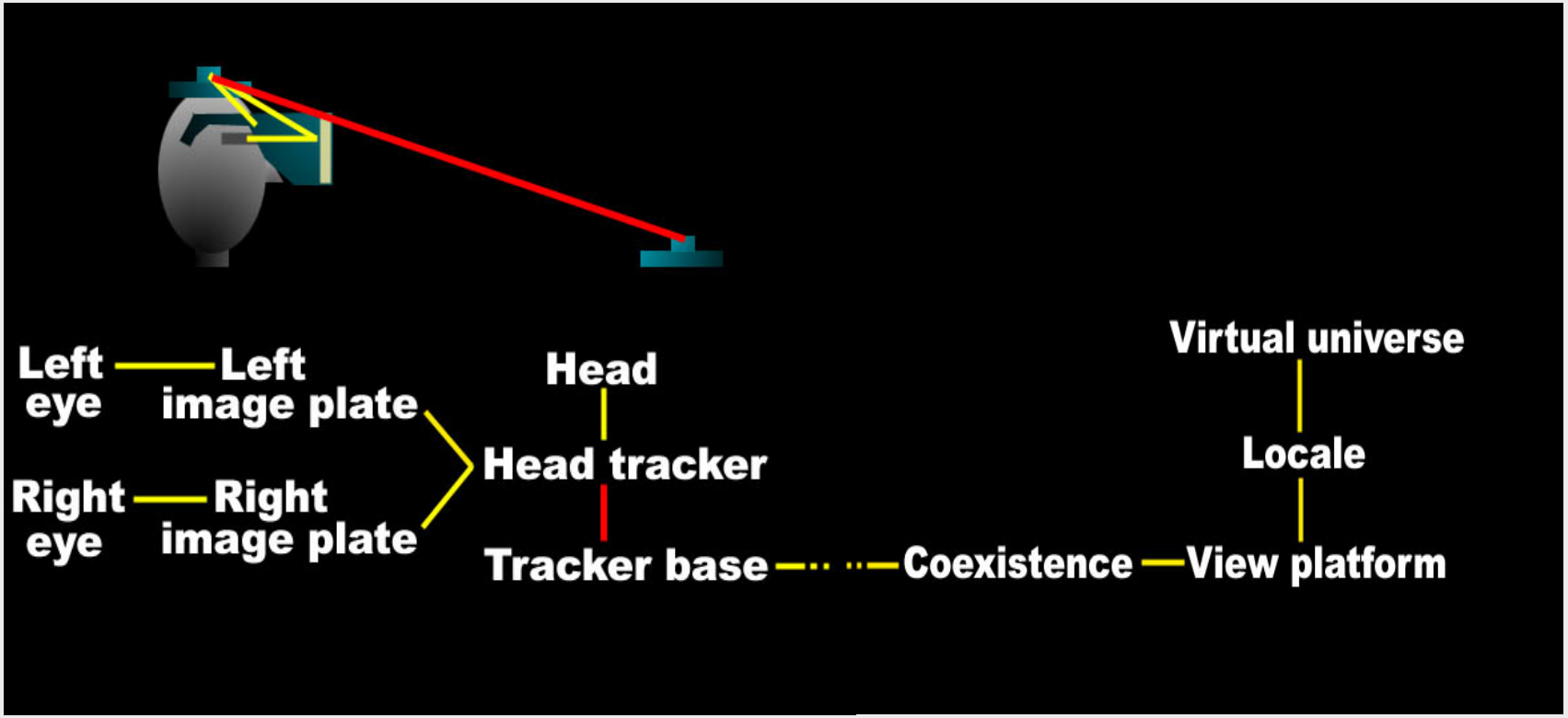
Understanding physical to virtual mappings

- On the virtual side, the scene graph establishes:
  - Mappings from view platform center, to locale, to virtual universe
  - The view platform's center *co-exists* with the center of the portal (or wherever the coexistence transform selects)
- Together, these physical and virtual mappings establish coexistence
  - Movement in the physical world gives proper corresponding movement in the virtual world

Room-mounted display



Head-mounted display



Looking at view model classes

- Let's look at which classes are involved in the view model
- A VirtualUniverse defines the universe coordinate system
- A Locale places a scene graph branch within that universe
- A ViewPlatform (and a Transform3D above it) defines a view point within that locale
  - It defines a frame of reference for the user's position and orientation in the virtual world
  - Think of it as a magic carpet
  - There can be many ViewPlatforms in a scene graph

Looking at view model classes

- A View is the virtual user standing on a ViewPlatform
  - There can be many Views on the same ViewPlatform
- A PhysicalBody describes the user's dimensions for use by a View
  - There is always one PhysicalBody for a View
- A PhysicalEnvironment describes the user's environment for use by a View
  - There is always one PhysicalEnvironment for a View

Looking at view model classes

- A Canvas3D selects a screen area on which to draw a View
  - Every View has one or more Canvas3Ds
- A Screen3D describes the physical display device (image plate) drawn onto by a Canvas3D
  - A Canvas3D always has a Screen3D to draw onto
- The virtual user's location and orientation is controlled by a ViewPlatform:
  - A Transform3D above the ViewPlatform moves the platform about
  - The *view attach policy* aligns the platform origin with the user's screen, head, or feet

Looking at what is where

- Viewing policies and parameters are controlled by a *View*
  - The projection policy selects perspective or parallel projection
  - The view policy selects the room- or head-mounted display constraint systems
  - Various window policies control how the view frustum adapts to viewing parameter changes

Looking at what is where

- The user's physical dimensions are described by a PhysicalBody
  - Parameters set the left and right eye and ear positions
  - Parameters also set the nominal head height from the ground, and the nominal eye offset from the nominal screen
  - A transform describes the head to head tracker relationship

Looking at what is where

- The user's display, input sensors, and sound environment are described by a PhysicalEnvironment
  - A transform describes the coexistence to tracker base relationship
  - A set of abstract input sensors provide access to trackers
  - An audio device enables sound playback

Looking at what is where

- The drawing area is selected by a Canvas3D
- The physical screen device is described by a Screen3D (image plate)
  - A transform describes the tracker base to image plate relationship
  - Parameters set the display's physical width and height (in meters)

Summary

- Virtual world:
  - ViewPlatform controls the user's virtual position and orientation
  - View sets the view policy, etc.
- Physical world:
  - PhysicalBody describes the user
  - PhysicalEnvironment describes the user's environment
  - Canvas3D selects a region to draw into
  - Screen3D describes the screen device

Configured Universe

27/04/2015

ConfiguredUniverse

- Utility class creates all necessary objects to set up the view side on a scenegraph
  - Locale
  - One or more viewing platforms
  - At least one Viewer object
- ConfiguredUniverse can set up a viewing environment based upon the contents of a configuration file
  - This allows an application to run without change across a broad range of viewing configurations
    - Desktop, fullscreen, stereo enabled views, cave, HMD, 6DOF sensorer …

ConfiguredUniverse

- A configuration file may create InputDevice, Sensor, and ViewPlatformBehavior instances as well as Viewers and ViewingPlatforms
- If a configuration file or container is not provided.
  - then ConfiguredUniverse creates a default viewing environment in the same way as SimpleUniverse
- The syntax and description of the configuration file may be found in Suns Java3D javadoc
- Several example configuration files can be found in Suns Java3D javadoc