CSE 3200 Micro-Computer Graphics
Coordinate Systems – Spaces

Presenter: Girendra Persaud
University of Guyana

Outline

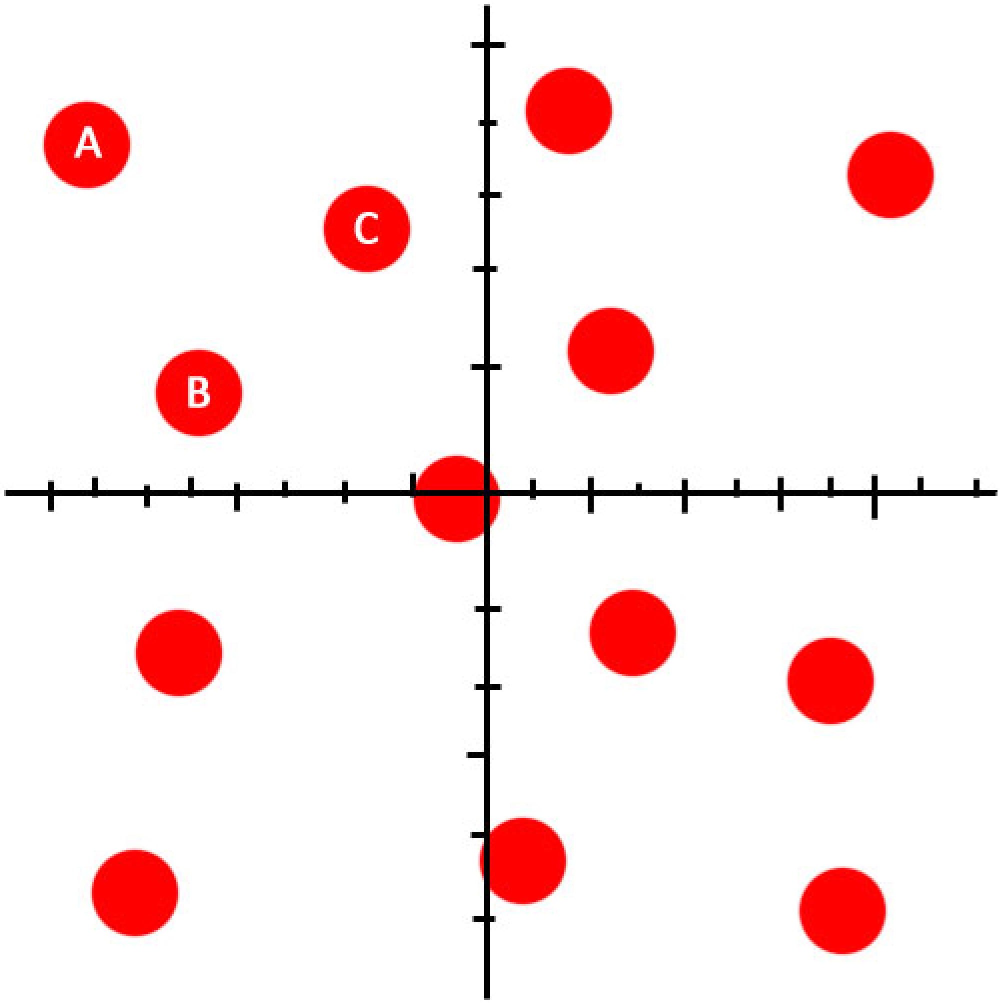
- Definition of a point
- Abstract coordinate System
- Spaces (Modeling)
  - Object Space
  - World Space
  - Camera Space
- Vertex data to pixel data (rasterization)
  - GL_MODELVIEW
  - Clip Coordinates
  - NDC
  - Windows Coordinate
- Conclusion
- Questions?
- Review Questions

What is a point?

- Something with
  - No dimensions
  - Has location, position

- How is a point referenced?
  - That point over there?
  - The point to the left of that point?

Referencing



Abstract Coordinate System

A coordinate system that exist only in relation to what is defined, then and there and at no other time. Only in that instance.

Spaces

- There are three spaces that make up the overall geometry of a modeling system
  - Object space
  - World space
  - Camera Space

Object Space

- Known as model space
- Usually, but not always, each object will have its own distinct object space with the origin at the object's center
- Objects in its own object space will have positions and orientations relative to other objects in the hierarchy of that object

http://www.fastgraph.com/help/object_space_and_world_space.htm
l

World Space

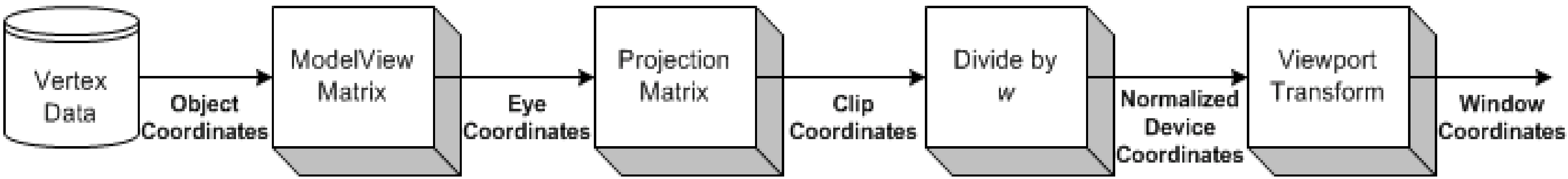
- Our 3D universe
- All objects in your scene are located in world space by their position, rotation, and scale
- Its central origin is the central origin about which rotation and scaling transformations of the entire scene
- World space represents possibly the highest point in the hierarchy
- Stationary objects such as walls are well-suited for definition in world space, but moving objects are best defined in object space
- World space coordinates are also used to define POV
  - By default, the POV is at the world space origin, looking straight down the positive z axis

http://www.fastgraph.com/help/object_space_and_world_space.html

Camera Space

- Also know as eye space
- It is the subjective view of the world and it is relative to world space
- It is the last transformation necessary before the image is put to the screen (the final stage of image projection).
- A +5 translation in the z axis, results in a multiplication of the camera matrix of the opposite magnitude (-5 in the z axis)
- gluLookAt (eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , upY , upZ);

Journey to the screen



Projection transformation is applied to the transformed scene in camera space then the scene is taken through a further set of operations before it is displayed on the final output (screen)

GL_MODELVIEW

- **Note: GL_MODELVEIW** matrix is a combination of Model and View matrices (M_{view} . M_{model})
- Model transform is to convert from object space to world space
- View transform is to convert from world space to eye space

$$\begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix} = M_{modelView} \cdot \begin{pmatrix} x_{obj} \\ y_{obj} \\ z_{obj} \\ w_{obj} \end{pmatrix} = M_{view} \cdot M_{model} \cdot \begin{pmatrix} x_{obj} \\ y_{obj} \\ z_{obj} \\ w_{obj} \end{pmatrix}$$

Clip Coordinates

- It is after applying eye coordinates into GL_PROJECTION matrix
- Objects are clipped out from the viewing volume (frustum)
- Frustum is used to determine how objects are projected onto screen (perspective or orthogonal) and which objects or portions of objects are clipped out of the final image

$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = M_{projection} \cdot \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix}$$

Projections

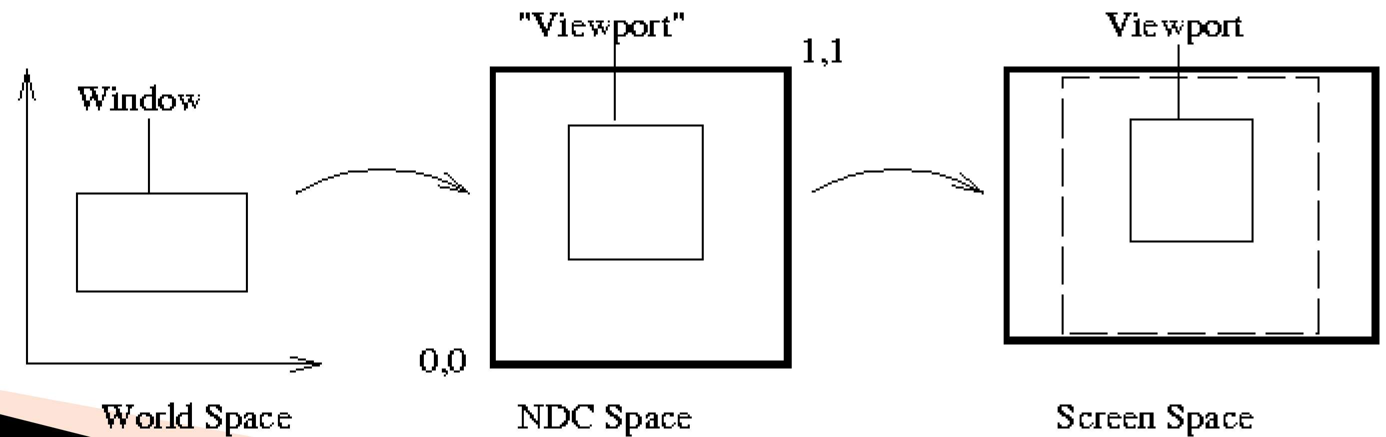
- **Orthographic**
  - glOrtho(left, right, bottom, top, near, far);
  - gluOrtho2D(left, right, bottom, top);

- **Perspective**
  - glFrustum(left, right, bottom, top, near, far);
  - gluPerspective(fov, aspect, near, far);

Normalized Device Coordinates

- An intermediate coordinate system that gets mapped to the device layer
- It is like window (screen) coordinates, but has not been translated and scaled to screen pixels
- The range of values is now normalized from -1 to 1 in all 3 axes

$$\begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} x_{clip}/w_{clip} \\ y_{clip}/w_{clip} \\ z_{clip}/w_{clip} \end{pmatrix}$$



Window Coordinates

- Calculated by applying normalized device coordinates (NDC) to viewport transformation
- The window coordinates finally are passed to the raterization process of OpenGL pipeline to become a fragment
- glViewport() command is used to define the rectangle of the rendering area where the final image is mapped
- glDepthRange() is used to determine the *z* value of the window coordinates
- The window coordinates are computed with the given parameters by 2 functions in OpenGL
  - glViewport(x, y, w, h);
  - glDepthRange(n, f);

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} \frac{\mathrm{w}}{2} x_{ndc} + (\mathrm{x} + \frac{\mathrm{w}}{2}) \\ \frac{\mathrm{h}}{2} y_{ndc} + (\mathrm{y} + \frac{\mathrm{h}}{2}) \\ \frac{\mathrm{f}-\mathrm{n}}{2} z_{ndc} + \frac{\mathrm{f}+\mathrm{n}}{2} \end{pmatrix}$$

Conclusion

- The Space:

object space >>> world space >> eye space >>clip space >> normalized device space > >> window space

- Transformation is converting from one coordinate system to another

Questions?

Review Questions

- What does it mean to have a hierarchical arrangement with articulated rigid bodies?
- In your project, what are the things you would define in world space?
- Why might NDC be necessary?