Outcomes

- Understand the creation and use of many of the basic components of FreeGLUT UI
- Be capable of constructing a viewing Frustum [illegible] with less parameters
- Work with built-in FreeGLUT objects

## Menus

- This describes how to use FreeGLUT to create menus activated by clicking the right mouse button → also called Context Menus
- Menu-related OpenGL calls
  - glutCreateMenu(menu_function) ≡ creates a menu, registers menu_function() as its callback. Returns a unique integer identifying the menu
    - the identifying integer can be used by any higher-level menu calling this one
  - glutAddMenuEntry(tag, returned_value) ≡ creates a new menu item
    - tag - menu label
    - returned_value - the value returned to the menu callback function when the menu item is clicked
  - glutAddSubMenu(tag, sub_menu) ≡ similar to glutAddMenuEntry()
    - tag - menu label that when clicked opens a new submenu
    - sub_menu - id value of the submenu that pops up
  - glutAttachMenu(button) ≡ attaches the menu to a mouse button
    - button - one of GLUT_RIGHT_BUTTON, GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON
- Note: Due to the nature of the calls menus must be created in a bottom-up fashion

Ex: Modify menus.cpp to add two more items to the top-level pop-up menu.

1. A "mode" option which allows the rectangle to be "Outlined" or "Filled"
2. A "size" option which leads to two sub-menu options "Width" and "Height" both of which have options "Small", "Medium" and "Large"

## Line Stipples

- Stippled Lines ≡ broken lines → - - - - - - → which can be created by specifying a stippling pattern

- Enabling / Disabling
  - To enable call glEnable(GL_LINE_STIPPLE)
  - To disable call glDisable(GL_LINE_STIPPLE)

- Specifying the Stipple Pattern
  - call glLineStipple(factor, pattern)
    - pattern: a hex string of the form $0x X_3 X_2 X_1 X_0$ where each $X_i$ is a hexidecimal symbol (aka 4 bits)
    - factor: a positive integer defining the number of pixels on/off in each group specified by the bits in pattern



- Note: FreeGLUT stroke characters can also be stippled

Ex: Apply different line stipples from lineStipple.cpp to the circle in circle.cpp

Ex: Display the text "OpenGL is Awesome!!" using various stippled stroke fonts.

## FreeGLUT Objects

- FreeGLUT offers a collection of pre-defined objects in both solid and wire-frame forms

| Object | Solid |
|---|---|
| Sphere | glutSolidSphere(radius, slices, stacks) |
| Cube | glutSolidCube(size) |
| Cone | glutSolidCone(base, height, slices, stacks) |
| Torus | glutSolidTorus(innRadius, outRadius, sides, rings) |
| Dodecahedron | glutSolidDodecahedron(void) |
| Octahedron | glutSolidOctahedron(void) |
| Tetrahedron | glutSolidTetrahedron(void) |
| Icosahedron | glutSolidIcosahedron(void) |
| Teapot | glutSolidTeapot(size) |

- Wireframe versions are the same calls with "Solid" replaced by "Wire"

Exp: Download and run glutObjects.cpp to see various versions of the FreeGLUT objects.

Use x/X, y/Y, and z/Z to turn them

## Clipping Planes

- We can specify clipping planes beyond those of the viewing volume with:

  `glClipPlane(GL_CLIP_PLANEi, *equation);`

  - This specifies the $i^{th}$ additional clipping plan
  - equation: points to an array $\{A, B, C, D\}$ which are the coefficients of the following equation:

    $$Ax + By + Cz + D = 0$$

    of the clipping plane

- If the plane is enabled with `glEnable(GL_CLIP_PLANEi)`

  ~~The points which lie in the~~
  - The points $(x, y, z)$ which lie in the open half-space:

    $$Ax + By + Cz + D < 0$$

    are clipped off
  - The points $(x, y, z)$ in the closed half-space:

    $$Ax + By + Cz + D \geq 0$$

    are rendered

- The plane may be disabled via `glDisable(GL_CLIP_PLANEi)`

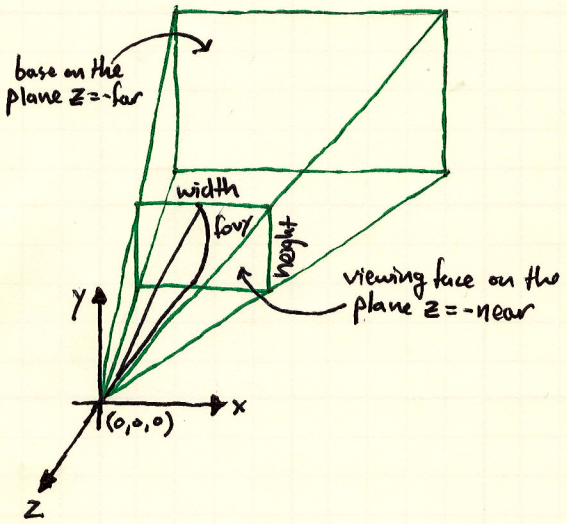
## Frustum Freely

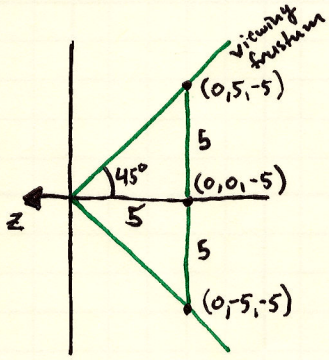
- we can use the call

  gluPerspective(fovy, aspect, near, far)

  as an equivalent to glFrustum() method to construct a frustum view. where:

  - fovy: defines the field of view angle, which is the angle subtended along the yz-plane at the apex of the pyramid
  - aspect: is the aspect ratio = width/height of the front face of the frustum
  - near and far: remain the same




- These 4-parameters are enough to determine the eight parameters needed by glFrustum() where left=-right and bottom=-top (aka symmetric about the z-axis)

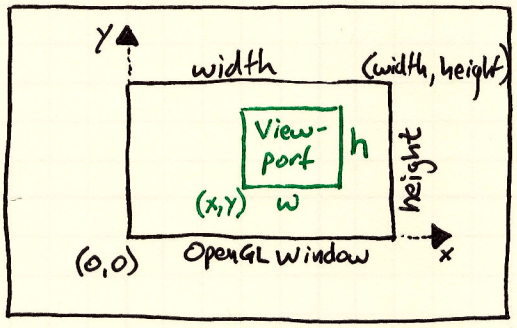
**Ex:** Determine the equivalent glFrustum() call of the following projection statement:

gluPerspective(60.0, 2.0, 10.0, 100.0)

**Hint:** Use trigonometry in the yz-section to first determine the top and bottom values and then the aspect ratio to determine left and right

## Viewports

- Viewport ≡ the region of the OpenGL in which a scene is drawn
  - By default, this is the whole window
  - glViewport() can be used to draw a smaller rectangular subregion




glViewport(x, y, w, h)

- glViewport(x, y, w, h) - specifies the viewport as the rectangular subregion of the OpenGL window
  - (x,y) lower-left corner location
  - w - width
  - h - height
- Multiple viewports can be created in a single window by invoking more than one glViewport()
  - The contents of a particular viewport is defined by the statements following the glViewport() call and preceding the next one.
  - Useful in games to show split-screens or the same scene from different angles
- glutCreateWindow() may be invoked more than once to create multiple top-level windows

Ex: Create 3 top-level windows with red, blue, and green backgrounds and containing the words "Red", "Blue", and "Green"

## For Next Time

- Read Chapter 4 sections 1-5
- Review these lecture notes
- Come to class!
- Work on HW 01

## Additional Notes