1. Short Answer / Multiple Choice / True or False (5 points each)

1A. When accessing a mouse sensor, what types of data are available? (Circle all that apply.)

a. Manufacturer's model number.

b. Position data and button data.

c. Raw data and miscellaneous data.

d. Direction and distance to nearest cheese.

e. None of the above.

1B. True or False: Level of Detail nodes can be used to provide animation to a simulation.

a. True

b. False

1C. A motionlink uses position and orientation information from a *source* to affect the position and orientation of a *target*. In the table below, indicate whether the given item is a valid source, target, or both by placing a "Y" in the appropriate cells.

Item	Valid Source?	Valid Target?
Movable Node		**Yes**
(Mouse) Sensor	**Yes**	
WTpq Variable		
Path	**Yes**	
Viewpoint		**Yes**

1D. Given the material table below, select the material ID that is most suitable for each of the following objects:

a. Grass (i.e. golf course lawn, not a controlled smoking material.) **0 (or 5)**

b. A brand new sports car. **2**

c. A 2-liter 7-up bottle. **6**

d. Tubing for a neon sign **4**

e. A pool (billiards) table. **5 (or 0)**

```
mat
version 3.00
valid ambient diffuse specular shininess opacity emissive opacity


matdef // # 0
ambientdiffuse 0.0 0.8 0.0


matdef // # 1
ambientdiffuse 0.4 0.4 0.4


matdef // # 2
ambientdiffuse 1.0 0.0 0.0
specular 1.0 1.0 1.0
shininess 96.0 // 75% of maximum


matdef // # 3
ambientdiffuse 0.0 0.0 0.0


matdef // # 4
ambientdiffuse 0.0 1.0 0.0
emissive 0.0 1.0 0.0


matdef // # 5
ambientdiffuse 0.5 1.0 0.5


matdef // # 6
ambientdiffuse 0.0 1.0 0.0
opacity 0.5


matdef // # 7
ambientdiffuse 0.5 0.5 0.5
opacity 0.8
```

2. Geometry Modification - (10 points)

In the following code segment, it is desired to create snow-capped mountains, with foliage below the snow line (10,000 feet for this simulation.) A geometry file is available that has the correct shape, size, position, and orientation, but which does not have the correct color and texture.

Add code below the given segment to loop through all the polygons in the terrain, and <u>either</u> color the polygon pure white if its center of gravity is above 10,000 feet (i.e. the Y component is less than –10000), <u>or</u> apply the texture "FOLIAGE.TGA" otherwise. All the variables you need should be present in the given segment, however you may also declare additional variables if you wish.

```
WTp3 cg;
WTgeometry *geo = NULL;
WTpoly *poly = NULL;
WTnode *terrain = NULL;


terrain = WTnode_load( Root, "TERRAIN.NFF", 1.0 );

/* Add your code here to decorate the terrain */

geo  =  Wtnode_getgeometry(  terrain  );

for(  poly  =  WTgeometry_getpolys(  geo  );  poly;
      poly  =  WTpoly_next(  poly  )  )  {

      WTpoly_getcg(  poly,  cg  );

      if(  cg[  Y  ]  <  -10000  )
           WTpoly_setrgb(  poly,  255,  255,  255  );
      else
           WTpoly_settexture(  poly,  "FOLIAGE.TGA",
                FALSE,  FALSE  );

}
```

3. Concept Explanation and Distinction - (20 points)

A. In WTK, what is a "path", and how is it used? Include in your explanation an example of how you might use a path in your project. (You may place all or part of your answer on a separate page if you wish. See also part B below before completing part A.)

A path is an ordered collection of (position, orientation) information. It is usually connected to a movable node or the viewpoint in order to move them along a given route. An example would be an escalator, which moves a person from one floor to another.

B. Repeat part A for "nodepath" as opposed to "path".

A nodepath is a sequence of nodes in a scene graph, from an ancestor (usually Root) to a descendant (usually a "leaf" node.) It is used to distinguish a particular instance of an object on the scene graph. An example is picking an object with the mouse - WTscreen_pickpoly returns a nodepath.

4. Original Code Development - (50 points total)

In the first mid-term exam, a merry-go-round carousel was added to a scene graph, along with four horses. We now want to add motion to the simulation by creating two task functions – One to rotate the merry-go-round, and one to move the horses up and down on their poles. You may want to read all sections of this question and review the solution pages from the first mid-term (which start on page 7) before beginning.

A. **(5 points)** Write the function prototypes for two task functions, one to rotate the merry-go-round and the other to position the horses up and down.

```
void  rotate_carousel(  WTnode  *  );

void  raise_horse(  WTnode  *  );
```

B. **(10 points)** In the code segment below (excerpted from the exam 1 solution), add code in the indicated areas to assign tasks to the carousel and to each of the horses. Set the priorities such that the carousel task always goes first, followed by each of the horse tasks in order. (The first horse loaded should go first, followed by the second horse loaded, and so on.)

```
WTnode *Merry = NULL;

void loadMerryGoRound( WTnode *root ) {

     int i;
     WTnode *node = NULL, *sep = NULL, *sep2 = NULL;

     /* Code begins here */

     sep = WTsepnode_new( root );
     Merry = WTxformnode_new( sep );

     /*  Add  code  here  to  attach  a  task  to  "Merry"  */

     WTtask_new(  Merry,  rotate_carousel,  1.0f  );

     WTnode_load( sep, "PLATFORM.NFF", 1.0f );

     for( i = 0; i < 4; i++ ) {

          sep2 = WTsepnode_new( sep );
          node = WTxformnode_new( sep2 );

          /*  Add  code  here  to  attach  a  task  to  "node"  */

          WTtask_new(  node,  raise_horse,  2.0f  +  i  );



/* Remainder of function ommitted */
```

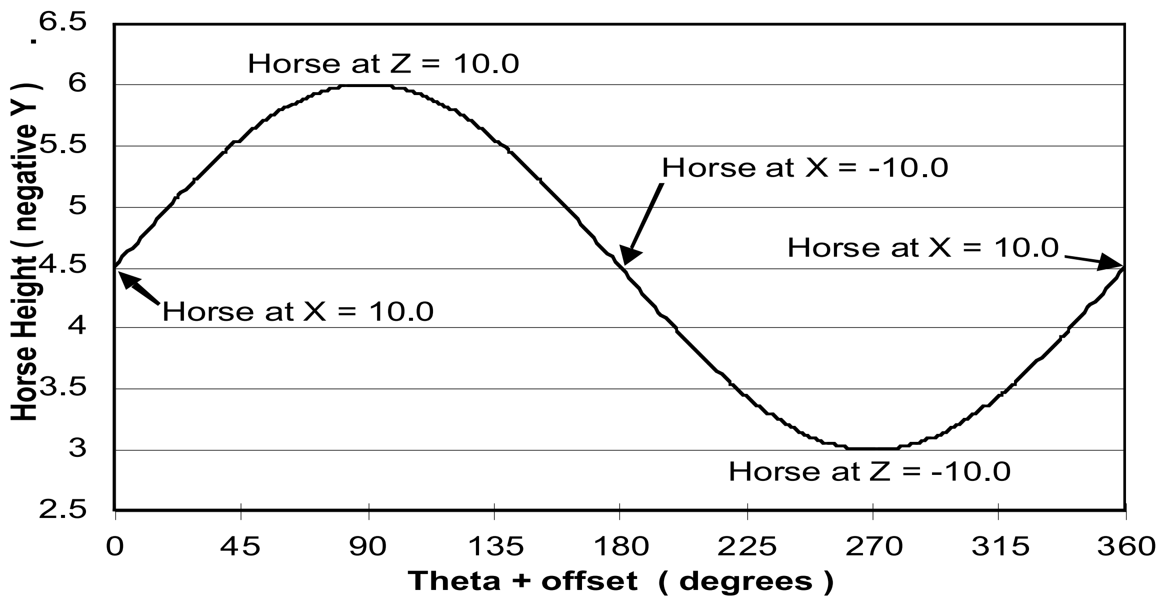
C. **(15 points)** On a separate sheet, write the task function to rotate the carousel. Note the following points:

- Although the exam 1 solution creates a global variable "`Merry`", that variable should not be used by the task function.
- The carousel task function should use and update the global variable "`Theta`", declared as a `float` and initialized to 0.0, which maintains the current rotation of the carousel.
- The carousel task function should also reset the global integer "`WhichHorse`" to zero.

```
/***************************************************************/
#define DELTA 0.314159 /* Some small value.  e.g. Pi / 10 */
void rotate_carousel( WTnode *carousel ) {
    WTq orientation;  /* Only needed for alternate answer */
    Theta += DELTA;
    WhichHorse = 0;
    /* Easy Answer: */
    WTnode_axisrotation( carousel, Y, -DELTA, WTFRAME_LOCAL );
    /* Alternate Answer:  Note this one is commented out. */
    /* WTeuler_2q( 0.0f, -Theta, 0.0f, orientation );
         WTnode_setorientation( carousel, orientation ); */
    return;
}
```

D. (**20 points**) On a separate sheet, write the task function to elevate each horse. Note the following points:

- Although the exam 1 solution creates global variables for the horses, those variables should not be used by the task function.
- The same task function will be used for all four horses. I.e. you are to write one task function, not four. (Use the global variable "`WhichHorse`"; See below.)
- The horse task function should set the height of the horse between –3.0 and –6.0, using a sin function, the global variable "Theta", and an offset to Theta of (90 degrees * the global variable "WhichHorse"). I.e. the first horse will be at – (4.5 + 1.5 * sin(Theta)), the second horse will be at – (4.5 + 1.5 * sin(Theta + π / 2)), etc. It is your choice whether to keep Theta in degrees or radians, but the math function "`sin`" uses radians.



- The horse task function should increment the global integer "`WhichHorse`" just prior to returning.

```
/*************************************************************/
#define HALF_PI ( 3.14159 / 2 )
void raise_horse( WTnode *horse ) {
    WTp3 translation;
    WTnode_gettranslation( horse, translation );
    translation[ Y ] = -( 4.5 + 1.5 * sin( Theta + WhichHorse
         * HALF_PI ) );
    WTnode_settranslation( horse, translation );
    whichHorse++;
    return;
}
```

Reference: Problem 3 Question and Solution from Exam 1

Given the scene graph and diagrams on the following pages, write the code for a function to load a simple merry go round into a WorldToolKit simulation. Additional information:

1. The WTnode *root is passed into the function as an argument.

2. In the scene graph, "S" indicates a separator node, "T" a transform node, "LOD" a level-of-detail node, "G" the platform geometry node, and "H" a horse geometry node.

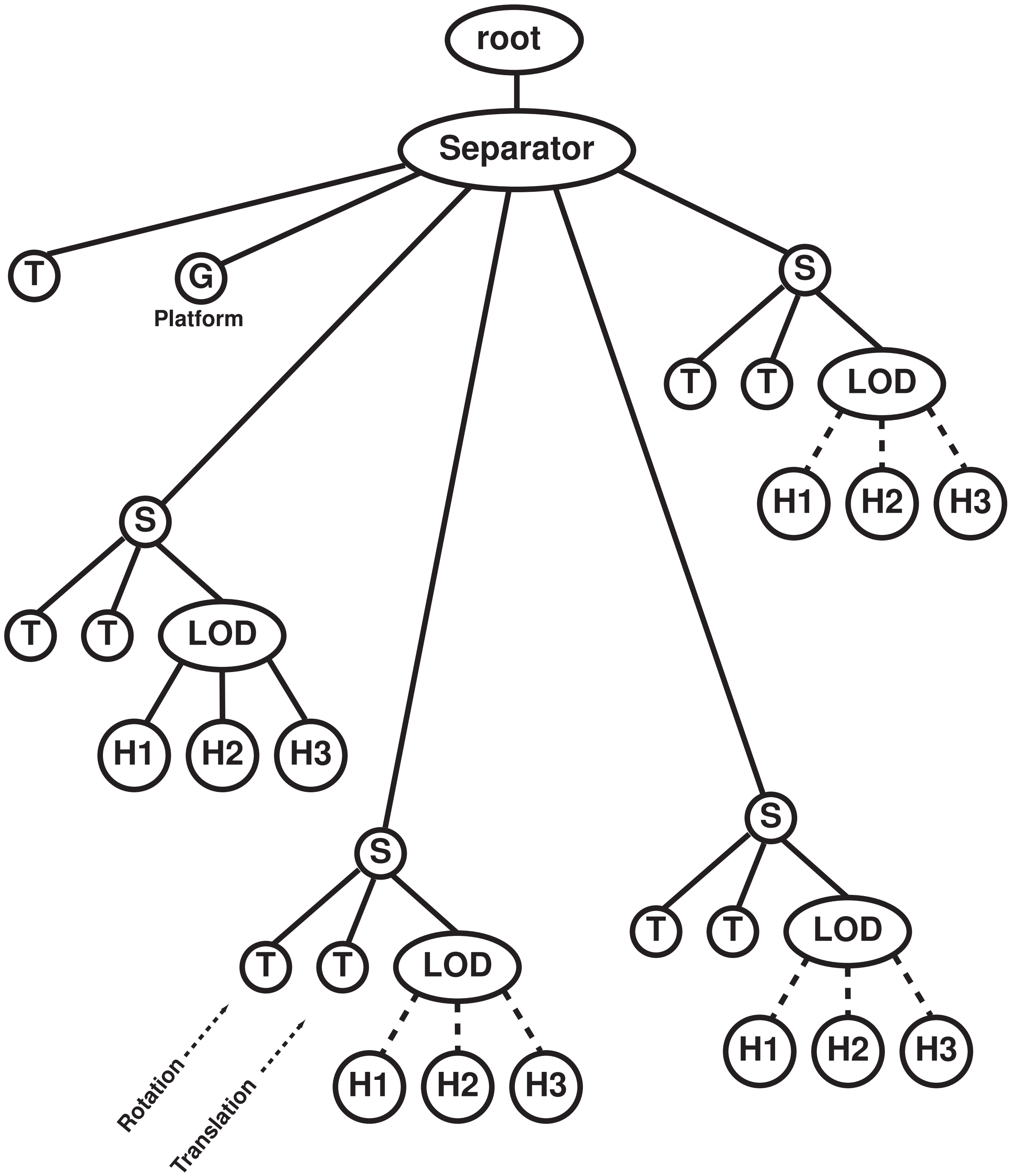
3. The first transform node places the entire merry-go-round in place, and rotates the merry-go-round as the simulation runs.

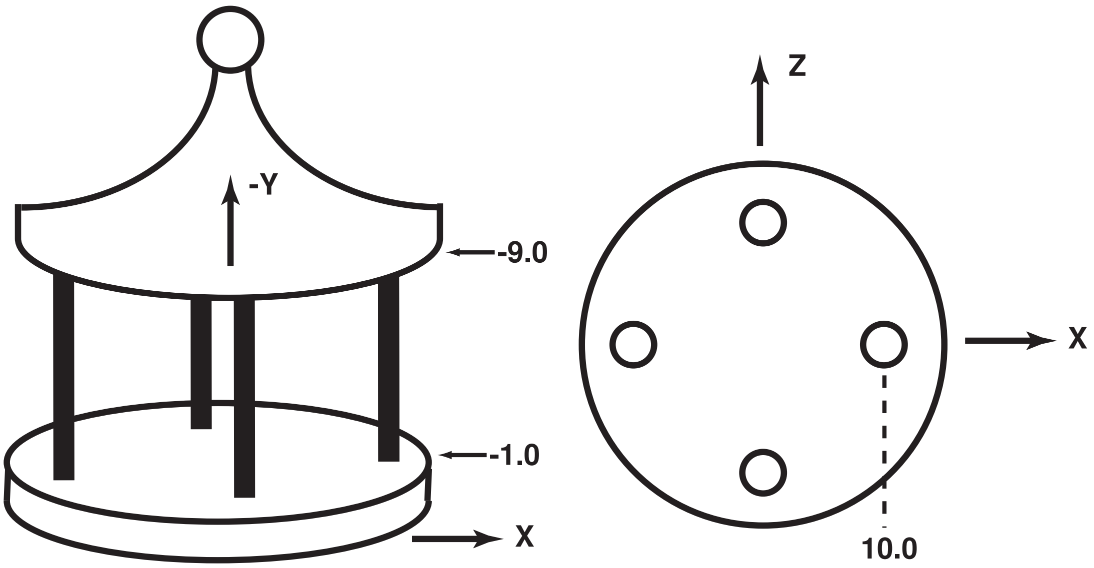
4. The platform geometry for the merry-go-round is located in the file PLATFORM.NFF. The platform has four poles, arranged as shown in the diagram, which remain stationary with respect to the platform. More specifically, the poles are 10 units away from the origin, along the positive and negative X and Z axes. The four horses will slide up and down on these poles.

5. Each horse has two transform nodes. The first rotates the horse for it's proper orientation. The second translates the horse from the origin to it's pole. As the simulation runs, the horses will move up and down, in the range from -3 to -6. Initially, the horses should be placed at different heights within this range, as indicated on the diagram.

6. The horse geometries are located in three files, HORSE1.NFF, HORSE2.NFF, and HORSE3.NFF, which are decreasingly detailed versions of the same horse. (I.e. HORSE1 is more detailed than HORSE2, which is more detailed than HORSE3.) Each of these geometries should only be loaded into memory once, and placed on the scene graph in as many locations as is necessary.

7. Feel free to annotate the scene graph with additional labels, if it will make your code easier to write and / or follow.

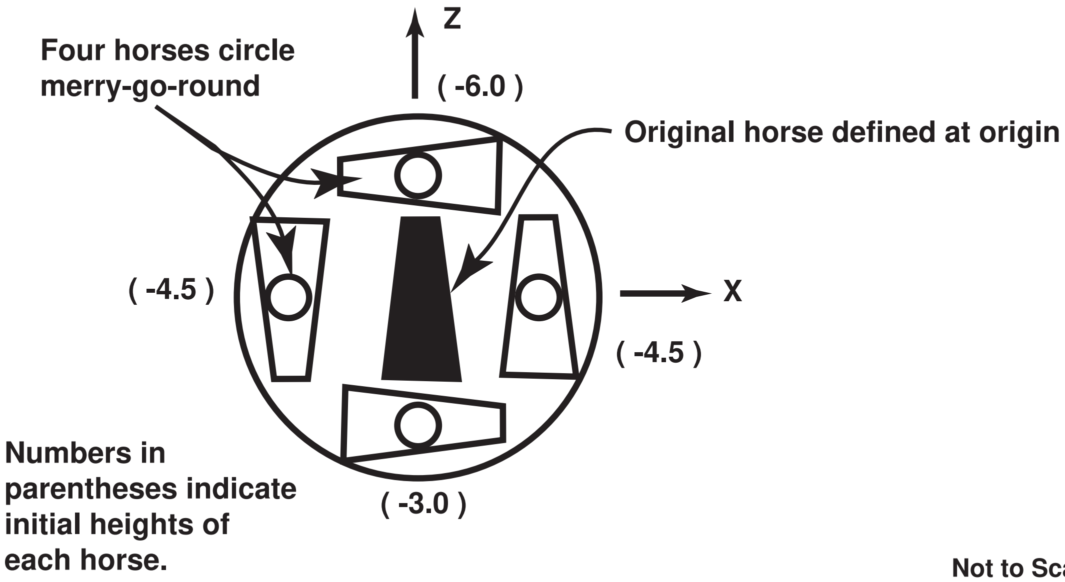

root
Separator
T
G
Platform
S
T
T
LOD
H1
H2
H3
S
T
T
LOD
H1
H2
H3
S
T
T
LOD
H1
H2
H3
Rotation
Translation
S
T
T
LOD
H1
H2
H3



PLATFORM.NFF

Not to Scale

HORSE*.NFF: A carousel horse, centered about the origin, facing forward along the positive Z axis. The length, width, and height of the horse are 4 units, 1 units, and 2 units respectively. I.e. the X coordinates of the horse extend from -0.5 to 0.5; the Y coordinates from -1.0 to 1.0, and the Z coordinates from -2.0 to 2.0.



Not to Scale

```
/* Declare Global Variables Here */

WTnode *Merry = NULL, *Horses[ 4 ] = { NULL, NULL, NULL,
NULL };
```

```
void loadMerryGoRound( WTnode *root ) {

     /* Declare local variables here */

     int i;

     WTnode *node = NULL, *sep = NULL, *lod = NULL, *sep2 =
     NULL,
              *horse1 = NULL, *horse2 = NULL, *horse3 = NULL;

     WTp3 offsets[ 4 ] =
          { {  10.0f, -4.5f,   0.0f },
            {   0.0f, -6.0f,  10.0f },
            { -10.0f, -4.5f,   0.0f },
            {   0.0f, -3.0f, -10.0f } };

     WTq rotations[ 4 ] =
          { { 0.0f,  0.0f,     0.0f, 1.0f    },
            { 0.0f, -0.7071f, 0.0f, 0.7071f },
            { 0.0f,  1.0f,     0.0f, 0.0f    },
            { 0.0f,  0.7071f, 0.0f, 0.7071f } };

     WTp3 center = { 20.0f, -5.0f, 30.0f };

     float lodRanges[ 3 ] = { 20.0, 50.0, 1.0e6 };


/* Continue local variables on additional page(s) as necessary */
     /* Begin code on a fresh page */
```

```
    /* Code begins here */

    sep = WTsepnode_new( root );
    Merry = WTxformnode_new( sep );
    WTnode_load( sep, "PLATFORM.NFF", 1.0f );

    for( i = 0; i < 4; i++ ) {

         sep2 = WTsepnode_new( sep );

         node = WTxformnode_new( sep2 );
         WTnode_setorientation( node, orientations[ i ] );

         Horses[ i ] = WTxformnode_new( sep2 );
         WTnode_settranslation( Horses[ i ], offsets[ i ] );

         lod = WTlodnode_new( sep2 );

         /* The following two lines were excluded from the
             exam question.  They are included in the
             solution only for completeness. */

         WTlodnode_setcenter( lod, center );
         WTlodnode_setrange( lod, lodRanges, 3 );

         if( i == 0 ) {

              horse1 = WTnode_load( lod, "HORSE1.NFF", 1.0f );
              horse2 = WTnode_load( lod, "HORSE2.NFF", 1.0f );
              horse3 = WTnode_load( lod, "HORSE3.NFF", 1.0f );

         } else {

              WTnode_addchild( lod, horse1 );
              WTnode_addchild( lod, horse2 );
              WTnode_addchild( lod, horse3 );

         } /* End of if-else block */

    } /* End of for loop through four horses */

} /* End of function to load merry go round */
```