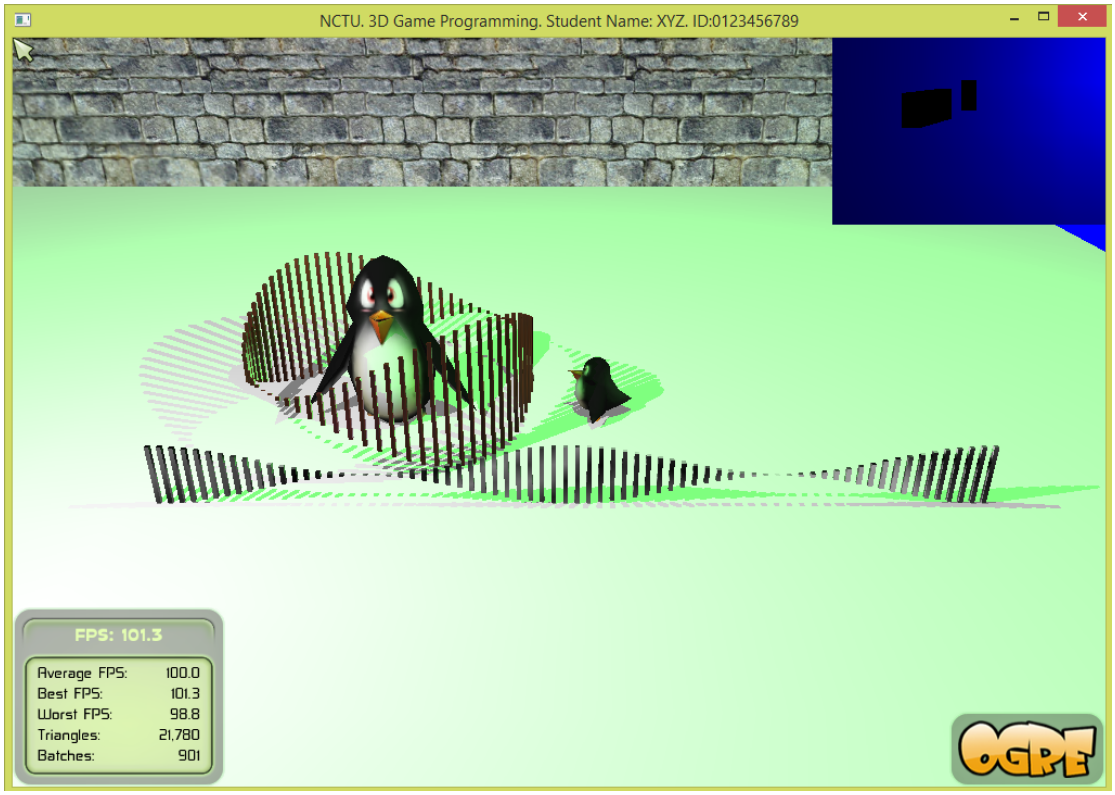
3D Game Programming: Programming Assignment One

YOU WILL RECEIVE ZERO POINTS IF YOU COPY and PASTE THE MATERIALS OF OTHER! LATE SUBMISSION WILL BE DEDUCTED BY 30 points a day.

In this assignment, you must implement the program individually. You should use the template to do this assignment. Rename the folder name to hw01_student ID.

You MUST implement something similar to the demo.



Penalties: **File organization is wrong [-20%]; program crashes [-20%], No student ID [-30%]. The program cannot be compiled on .NET2010 [-30%]. And something that is not similar to the demo.**

There are three tasks. You must finish all of them. The demo content may not be consistent with the instruction. Always follow the instructions to do the assignment. You must show two viewports with different contents as follows:

Task One [40%]:

1. [4%] Create a scene manager. Use it to create its own camera and the **first viewport**.
2. [4%] The viewport occupies the entire screen space.
   Background color must be PURE BLUE, i.e. (0, 0, 1).
3. [2%] Ambient light and shadow.
4. [4%] Create a plane and two penguins. Set the first penguin (large penguin) to look at the user. Set the second penguin (small penguin) to look at the first

penguin. The first penguin should be larger than the second penguin.

5. [10%] Create two set of objects by scaling cube objects. One set of objects form a circle around a large penguin. Another set of objects form a row.
6. [4%] Create two lights.
7. [2%] Enable shadows.
8. [4%] Show the mouse cursor.
9. [6%] Show your name and ID on the top bar of the window.

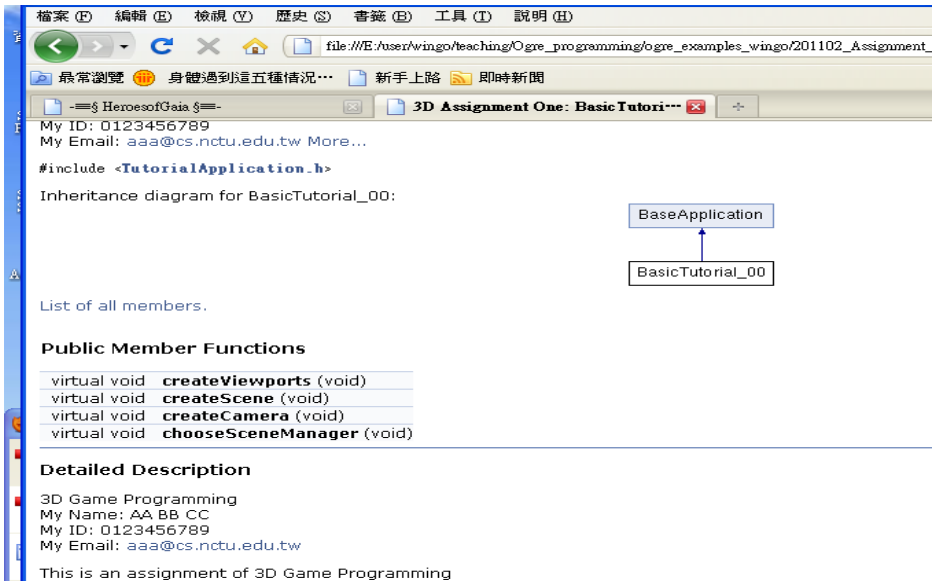
Task Two [20%]:

1. [2%] Create a scene manager. Use it to create its own camera and the **second viewport**.
   Set the camera position to (0,350,0.0). Set the camera to look at (0.0001, 0,0.0).
2. [2%] The viewport occupies the upper right corner.
3. [2%] Create a plane.
4. [2%] Ambient color.
5. [2%] One light with blue color (diffuse and specular).
6. [2%] Enable shadow.
7. [2%] Create an object, such as a sphere or a cube. Scale it with different scaling factors in x-, y- and z-directions. Must place it above the plane. Set it's x-coordinate to 50. **Its shadow must be seen**.
8. [4%] Edit your own material (pure green color). Apply the green material to the object (created in Task 7).
9. [2%] Disable overlays.

Task Three [20%]. If something is not clear, Check the demo for details.

1. [5%] Press 1, 2, 3, 4, and 5 to set the camera at different positions and directions.
2. [5%] Press 'M' to make the second viewport at the bottom level (z-order = 0) and the first viewport at the top level (z-order = 1). The first viewport is at the upper left corner. Disable the overlay in the second viewport**. Set the background color of the first viewport to other color (not blue).**
3. [5%] Press 'N' to make the first viewport at the bottom level (z-order = 0) and the second viewport at the top level (z-order = 1). The second viewport is at the right middle portion of the screen. Disable the overlay in the second viewport. **Set the background color of the first viewport to other color (not blue).**
4. [5%] Press '0' to toggle the animation of the large penguin at the circle. Move it up and down. The penguin is **accelerated** to move up and then fall down to the floor, and then repeat until the animation is stopped.

[20%] **Task Four**: Use doxygen to generate an on-line documentation browser for the program. You **MUST** document each function and each class that you implement. You **MUST** write down your name, student ID and email address in the TutorialApplication.h. Your name, student ID and email address must be shown in the documentation. You should document at least FOUR functions. The htm files must be stored in **./docs/html**. The entry htm file is index.html.



Please see the demo for details. You must do the same thing as the demo.

File organization:

The folder name must be hw01_STUDENT_ID. For example, if your ID is 123456789 and name is XYZ. The folder name must be hw01_123456789. Make sure that your folder hw01_STUDENT_ID must be organized as follows.

The root folder (hw01_123456789). Inside it, we have the followings:

\boost_1_42 : boost files

\include : header files

\lib : contains the precompiled libraries

\programs, inside \programs it contains \assign_01.

Inside assign_01, we have the following items:

\bin : contains the executable, materials, dll, etc.

\media :contain media files

\docs : documentation, such as \html generated by doxygen

\source : contains all the .cpp and .h

\release

.sln : the project file for .NET2010

Submission:

1. zip and upload your source code to the E3 platform.
2. Write a report and submit a hardcopy to describe the way how you finish the assignment. Submit the report in class. **No plagiarism in your report**. **If you copy the instruction to your report, your report score is ZERO. Write the report in your own words.**

Guideline. The guideline may not be consistent with the instruction. Always follow the instruction to do the assignment.

Task One:

1. **Create a scene manager. Use it to create its own camera and viewport.**
   The camera must be:

```
setPosition(Ogre::Vector3(120,300,600));
lookAt(Ogre::Vector3(120,0,0));
```

2. **The viewport occupies the entire screen space.**
3. **Ambient light and shadow:**

```
ColourValue( 0.5, 0.5, 0.5 )
mSceneMgr->setShadowTechnique(
        SHADOWTYPE_STENCIL_ADDITIVE);
```

4. Create a plane and two penguins.

To create a plane object:

```
Plane plane(Vector3::UNIT_Y, 0);
MeshManager::getSingleton().createPlane(
    "ground",
    ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME,
    plane,
    1500,1500,  // width, height
    20,20,      // x- and y-segments
    true,       // normal
    1,          // num texture sets
    5,5,        // x- and y-tiles
    Vector3::UNIT_Z // upward vector
    );
```

```
Scale up the first penguin by a factor (2, 3, 2).
Set the position of the first penguin at (0, 50, 0).
Make the second penguin face to the first penguin. The position of
the second penguin is (150, 20, 0).
```

5. Create two set of objects by scaling cube objects. One set of objects form a circle around a large penguin. Another set of objects form a row.

Scale the cubes accordingly.

The cube model: `cube.mesh`

To model a circle: PI = 3.141592654

```
numCubes = 72;
L = 255;
```

```
for (int i = 0; i < numCubes; ++i) {
    String name;
    genNameUsingIndex("c", i, name);
    Entity *ent = mSceneMgr->createEntity(name, "cube.mesh");
    ent->setMaterialName("Examples/SphereMappedRustySteel");
    AxisAlignedBox bb =  ent->getBoundingBox();
    cubeSize = bb.getMaximum().x - bb.getMinimum().x;
    x = 0, y = 0, z = -125;
    SceneNode *snode = mSceneMgr
           ->getRootSceneNode()
           ->createChildSceneNode();
    snode->attachObject(ent);
    fx = i/(double) (numCubes-1); // in range [0,1]
    h = (1+sin(fx*PI*4))*50; // height
    circle_radius = 100;
    x1 = circle_radius*cos(fx*PI*2);
    z1 = circle_radius*sin(fx*PI*2);
    unitF = 1.0/cubeSize/numCubes*L*0.8;
    snode->scale(unitF, h/cubeSize, unitF);
    snode->setPosition(x1, 50, z1);
}
You MUST use
setMaterialName("Examples/SphereMappedRustySteel")
and apply it to all the objects in the circle.
```

To model a row of objects, we use the height function:

```
fx = 2*i/(double) (numCubes-1); //i from 0 to numCubes-1
x = fx*L - L/2.0;
h = (1+cos(fx*3.1415*2.0))*20; // height
Real unitF = 1.0/cubeSize/numCubes*L*0.8;
snode->scale(unitF, h/cubeSize, unitF);
snode->setPosition(x, 20, z);
```

You MUST assign the material:

```
setMaterialName("Examples/Chrome");
```

to all the objects (modeled by cubes) in the row.

6. Create two lights.

```
light = createLight("Light1"); //error
light->setType(Light::LT_POINT);
light->setPosition(Vector3(150, 250, 100));
light->setDiffuseColour(0.0, 1.0, 0.0);
light->setSpecularColour(0.0, 1.0, 0.0);

light = createLight("Light2"); //error
setType(Light::LT_POINT);  //error
setPosition(Vector3(-150, 300, 250));  //error
light->setDiffuseColour(0.5, 0.5, 0.5);
light->setSpecularColour(0.5, 0.5, 0.5);
```

7. Enable shadows.

8. Show the mouse cursor.

In `void BaseApplication::createFrameListener(void)`

Use mTrayMgr->showCursor();

9. Show your name and ID on the top bar of the window.

Change the following in **bool BaseApplication::configure(void):**

```
mWindow = mRoot->initialise(true, "NCTU. 3D Game Programming.
Student Name: XYZ. ID:0123456789");
```

Task Two:

1. **Create a scene manager. Use it to create its own camera and viewport.**
   The position of the camera must be `Ogre::Vector3(0,350,0.001);`
   The camera must look at `Ogre::Vector3(0,0,0.0);`
2. **The viewport occupies the upper right corner.**
   `X = 0.0; Y = 0.0; Width =0.25; Height = 0.25`
   Set the background color.
3. **Create a plane.**
4. **Ambient color:** `ColourValue( 0.0,0.0, 0.0 )`
5. **One Light:**

```
light->setType(Light::LT_POINT);
light->setPosition(Vector3(100, 150, 250));
light->setDiffuseColour(0.0, 0.0, 1.0); //blue
light->setSpecularColour(0.0,0.0, 1.0); //blue
```

6. **Shadow type:**

```
mSceneMgr->setShadowTechnique(
          SHADOWTYPE_STENCIL_ADDITIVE);
```

7. **Create an object such as a cube or sphere. Scale it and place it at the center and above the plane.**
8. **Edit your own material (pure green color).**

Add the material in ./media/materials/scripts/#Examples.material.

```
material Examples/green
{
        technique
        {
            pass
            {
                    ambient 0.0 0.0 0.0
                    diffuse 0.0 0.7 0.0
                    specular 0.0 1.0 0.0
            }
        }
}
```

9. **Disable overlays.**

```
Ogre::Viewport* vp
vp->setOverlaysEnabled(false);
```

Task Three

1. [5%] Press ‘1’, ‘2’, ‘3’, ‘4’, and ‘5’ to set the camera at different positions and directions. Check the demo for details.

   Implement in bool keyPressed(const OIS::KeyEvent &arg)

   ‘1’: position: (98.14, 450.69, 964.20); direction: (-0.01, -0.30 ,-0.95)
   ‘2’: position: (-1463.00, 606.45, -513.24); direction: (0.88, -0.47, 0.10)
   ‘3’: position: (-1356.16, 634.32, -964.51); direction: (0.71, -0.44, 0.55)
   ‘4’: position: (40.39, 155.23, 251.20); direction: (-0.02, -0.41, -0.91)
   ‘5’: position: (19.94, 822.63, 30.79); direction: (0.00, -0.99, -0.11)

2. [5%] Press ‘M’ to make the second viewport at the bottom level (z-order = 0) and the first viewport at the top level (z-zero = 1). The first viewport is at the upper left corner. Disable the overlay in the second viewport. Check the demo for details.

   You must remove the existing viewports first. For example:

```
mWindow->removeViewport(mViewportArr[0]->getZOrder());
 mWindow->removeViewport(mViewportArr[1]->getZOrder());
```

   After that create new viewports as usual.

3. [5%] Press ‘N’ to make the first viewport at the bottom level (z-order = 0) and the second viewport at the top level (z-zero = 1). The second viewport is at the right middle portion of the screen. Disable the overlay in the second viewport. Check the demo for details.

4. [5%] Press ‘0’ to toggle the animation of the large penguin at the circle. Move it up and down. The penguin is **accelerated** to move up and then fall down to the floor. Repeat the process until the animation is stopped. Check the demo for details.

Update the large penguin in

```
bool frameStarted(const Ogre::FrameEvent& evt)
```

Approach: Use a variable (or more variables) to keep track of the animation state of the large penguin keyPressed function. And then based on the variable to update the large penguin. The penguin velocity should also be stored properly. We need to use it in frameStarted function.

To fall down in a free fall style:
Vector3 acceleration = Vector3(0.0, -50.8, 0.0);
velocity += acceleration*time_step
position += velocity*time_step
If position.y < 0.0 then position.y = 0.0

To move up, we have
Vector3 acceleration = Vector3(0.0, 20.8, 0.0);
The maximum height is 280.
The maximum speed is 80.

Task Four: Use doxygen to make documentation of the program. Briefly describe each function and each class that you have implemented. The Mode should be “**Documented entities only**”. You should document at least FOUR functions. **Do not copy and paste the solutions of the teacher.**

Report format

Name: _____ Student ID: _____Assignment:_____

Email:_____

THIS MUST BE YOUR OWN WORK! YES (Please Tick Yes)

*****BONUS: The best report(s) has at most 13 bonus points.**

[10%] Introduction (At least 100 words**)**

WORD COUNT: _____ //must fill this blank.

// describe the purpose of this assignment

// describe the tasks that you have to finish

[10%] System architecture

-[5%] Draw a diagram of the system. At least FIVE components.

-[5%] Describe in words about the system. At least 50 words.

WORD COUNT: _____ //must fill this blank.

[30%] Methods (At least 300 words)

WORD COUNT: _____ //must fill this blank.

//describe how you finish the tasks one by one.

//You must state clearly how to implement each item!

[40%] Discussion (At least 400 words)

WORD COUNT: _____ //must fill this blank.

You must include the following items!

//1. Describe what you see on the screen, e.g. shadows, etc.

//2. Explain why the object in the second task is black.

//3.What happen if you change the position of the camera to `//Ogre::Vector3(0,350,0.0)? Why?`

//4. Try other different parameters and describe what you see or the effects.

//Extra bonus points will be given here if you could raise up some interesting ideas.

[10%] Conclusion (At least 100 words)

WORD COUNT: _____ //must fill this blank.

// what you have learnt, any problems, difficulties, the assignment tough or easy for you? What do you suggest for the next assignment?