CO1301: Games Concepts
Lab 4 - Parenting and Models

1. Introduction

1. In this lab we will look at how two objects (e.g. models and cameras) can be linked to form parent/child relationships. We will then look at how parenting operates within a model as well as between models.

2. There are a huge number of possibilities to this technique: for example, attaching a gun object to a character object, allowing a character to move around on a moving boat, or making a camera to follow a character around (chase view).

2. Parenting

1. We will start by linking two objects together to create a chase view. The method used for linking objects is:

   ```
   void AttachToParent(const ISceneNode* Parent);
   // E.g. camera->AttachToParent( character );
   ```

   The type 'ISceneNode' is new to you – it represents anything in the scene, including cameras and models. This simply tells you that you can attach anything to anything (although some combinations don't work yet).

2. Create a new TL-Engine project named Lab4_Parenting_Project. Open the project.

3. Load the grid mesh, the cube mesh and create a model for each of them at the origin.

4. Load the sphere mesh and create a sphere model at position (-30, 0, -10).

5. Add code to allow you to move the cube using the keyboard (using w, a, s, and d keys).

6. Now attach the sphere model to the cube model. Look at the example in the beginning of this section fro the line of code to use, the code should be added in the initialisation section. Now the sphere should follow the cube when it moves.

7. Now add code to allow you to move the sphere (using u, h, j, and k keys). Do you understand how it moves relative to the cube? It is a similar concept to the local movement you covered in a previous lab.

8. Now change the camera to a "kManual" type and then attach the camera to the cube (the kFPS type of camera will not work here). You should now have a basic chase view camera.

9. You can detach an object from its parent with:

   ```
   // Detaches this object from any parent it is attached to.
   void DetachFromParent();
   ```

10. Save your project.

3. WWI Airplane Model

1. This section will walk you through building up a simple airplane scene.

2. Create a new TL-Engine project and name it Lab4_Airplane_Project.

3. The models you will need for this prototype are: "Skybox 01.x", "Plane.x", "Floor.x" (and texture "ground_01.jpg"); and they can be found in "models.zip" on Blackboard. Unpack

the models and place them in a folder called "media" within your project folder. Use the AddMediaFolder method to tell TL-Engine to get resources from the folder.

4. Open your created project, then create a FPS camera, load "Plane.x" and create a model using it.

5. Load and create a "Floor.x" model.

6. Notice how the wheels of the plane are sunk into the floor. Change the initial height of the plane so that the wheels are sitting on the floor. Remember this height - it is ground level.

7. Load "Skybox.x" and create a skybox model. The clouds on the skybox won't look correct when it is positioned at the origin (try it and see) so position it at (0, -940, 0) instead.

8. Add keyboard controls to rotate the plane clockwise and anticlockwise on the floor.

9. Create a variable to store the speed of the plane. Add keyboard controls to accelerate and decelerate the plane. The plane speed cannot be less than 0, i.e. stationary. Add plane movement into the game loop so that the plane now moves according to its current speed.

10. Using the AttachToParent command to create a chase camera. The "parent" in this case is the plane and the camera is the "child". Then use MoveLocal to position the camera behind and slightly above the plane:

```
camera->MoveLocal( 0.0f, -8.0f, 14.0f );
```

11. The plane is actually made up of several parts. These are the body of the plane, the left wheel, the right wheel and the propeller. These parts are called nodes. It is possible to move the nodes independently. The data type of nodes is ISceneNode.

```
ISceneNode* IModel::GetNode( const int iNode );
//Example usage: ISceneNode* propNode = plane->GetNode( 4 );
```

Model nodes can be manipulated with the same set of functions as models and cameras, e.g.

```
propNode->RotateLocalZ( 1.0f );
```

The above code will rotate the propeller of the plane. But you have to obtain the propeller node when you set up the scene first in order for this to work (the propeller is node number 4).

12. Now update your program so that the propeller of the plane starts rotating at a speed of 1.0f, but the rotating speed increases and decreases as the speed of the plane increases and decreases.

13. 'AttachToParent' creates a hierarchy of models. The child mirrors whatever happens to the parent. This is exactly what is happening with the nodes within a model. The nodes mirror what the parent does but can be manipulated independently. A model can be a hierarchy of nodes.

14. Allow the airplane to take off and to land. The plane should rise upwards when the speed is high enough (I used a value of 10). The plane should sink downwards when the speed is lower than this value. The plane should land if it reaches ground level.

4. Advanced Exercise

1. The front wheels of the plane should rotate when the plane is on the ground. Change your program so that the front wheels of the plane rotate when the plane is on the floor, but are stationary out when the plane is in the air.

2. The plane should bank when it turns. Rotate the plane model along its local Z axis in the correct direction when the plane turns.

3. Make the plane bounce slightly when it lands.