

Java3D Wind Turbine

Overview

My assignment was to create a 3D turbine using the Java3D API. The turbine must have the ability to rotate when the user clicks on a '3D button.' I also added some extra features and visual effects as discussed below.

I built the software on **Windows Vista** using **Java 6.0** and **Java3D 1.5.2**. I also tested it on Apple OS X 10.6.2. To run the software, double click on the **kxrs26-windmill.jar** file or type 'java -jar kxrs26-windmill.jar' in a console.

To see the source code, open the **Main/Main.java** file in the .jar file using ZIP-archive viewing software such as *WinRAR* or *unzip*.



Features

- **3D buttons** made from box primitives, designed to look like 2D buttons to the user. I applied an **image texture** to each button so that it is more obvious what each button does. The 3D buttons allow you to:
 - Adjust the wind speed up or down with a **smooth animation** (between an upper and lower speed limit)
 - Adjust the wind direction, both incrementally and automatically with a **smooth animation**
 - Change the wind direction
- A fixed **ambient light** and **coloured directional light** to add shading to the turbine.
- A **rotating directional light** which is applied when the automatic head rotation using the play (▶) button is pressed to **simulate a shadow being cast on the blades** (because Java3D does not support shadows)
- A rusty **image texture** applied to the stem of the turbine
- A **background image** of a snowy landscape
- The ability to **zoom** in and out using the mouse scroll wheel
- **Imported .obj files** containing the geometry of the wind turbine, created in the 3D modelling software, Blender

Java3D Features Used

Alpha
AmbientLight
Appearance
Background
BoundingSphere
Bounds
Box
BranchGroup
Canvas3D
DirectionalLight
Group
ImageComponent2D
Loaders (scene & ObjectFile)
Material
RotationInterpolator
SimpleUniverse
Texture
Transform3D
TransformGroup
OrbitBehaviour
PickTool, PickResult &
PickMouseBehaviour
TextureLoader

Controls

- ▲ **Increase the wind speed** by a fraction (makes the blades rotate faster)
- ▼ **Decrease the wind speed** by a fraction (makes the blades rotate slower)
- < **Change the wind direction** by a fraction to blow more towards the **left** if the automatic wind (see below) is not blowing already (rotates the turbine counter-clockwise)
- > **Change the wind direction** by a fraction to blow more towards the **right** if the automatic wind (see below) is not blowing already (rotates the turbine clockwise)

- ▶ **Create a wind** that automatically follows the turbine around (continually rotate the turbine)
- **Stop the wind** from following the turbine around (stop the turbine rotation)
- ↔ **Change the direction of the wind** which follows the turbine around (change the direction of the RotationInterpolator)

Implementation issues and their solutions

The most notable challenges I had to solve were:

- Rotating the turbine about the y-axis using the rotation interpolator was not in-sync with incremental rotations using the left (◀) and right (▶) buttons. To solve this I used a global variable to keep track of the current angle of rotation.
- Increasing or decreasing the wind speed using the up (▲) and down (▼) buttons caused a jump in the position of the blades. To solve this I set the minimum and maximum angles of the RotationInterpolator to be in-line with the value of the Alpha.
- It was not obvious what the 3D buttons did and the light animations were changing the shading which was off-putting. To solve this I applied an image texture to each button with an intuitive symbol and prevented them from reflecting light.
- Changing the direction of the automatic rotation of the head and blades caused a jump in the animation unless the change occurred when the RotationInterpolator was at either 0 degrees or 180 degrees due to the way in which the RotationInterpolator and Alpha work together. The further the rotation was from these two values at the time of the change, the larger and more noticeable the jump. To get around this, I created a second Alpha object which continually has the opposite value to the original Alpha, creating two more "safe" zones at 90 degrees and 270 degrees. Although there still is a jump if the direction is changed at certain angles, it is now two times less likely to happen and two times less noticeable if it does.
- Originally the user could speed up or slow down the wind speed as much as they liked. This looked bad at very high or low values and if you sped it up enough, it would start to slow down again due to the way the Alpha object works. To get around this, I set a minimum and maximum speed which cannot be exceeded by the user.