Sphere and light example

This example contains some neat stuff that you can utilize in your own projects in this course.

Configuring the application

We use `LwjglApplicationConfiguration` to configure the `LwjglApplication`. When Application configuration gets completed it can be better to use this class to group together the configuration and pass it to our `LwjglApplication` instance.

In this code we are fetching the desktop screen resolution and we use the same size for our application window. We then set the application to game mode. This means that the application will be executed in full screen. If your computer can't run 3D applications in fullscreen than change `cfg.fullscreen` to false. If the resolution does not work then alter the `cfg.width` and `cfg.height` to your needs.

Sphere

In this example we have new *complex* object that you can use in your programming assignment. The Sphere is implemented in a class named `Sphere`. This class can be used as follows.

```
Sphere sphere = new Sphere(10, 30);
sphere.draw();
```

This instance can be created in the `Create` and you can call the `draw` method from your rendering function. The first parameter is the slices and the second one are the stacks. This function is similar to `glutSolidSphere`.

You can decided if you Sphere should be displayed with lines or solid by a toggle function named `toggleDrawLines`.

Lights

In this example we are using three lights.

`GL_LIGHT0` is a diffuse light with the position `(30, 40, 15)` and color `(0.4, 0.5, 0.3)` as can be seen in the following line.

```
// Configure light 0
Gdx.gl11.glLightfv(GL11.GL_LIGHT0, GL11.GL_DIFFUSE, new float[
] { 0.4f,0.5f, 0.3f, 1.0f }, 0);
Gdx.gl11.glLightfv(GL11.GL_LIGHT0, GL11.GL_POSITION, new float
[] { 30.0f, 40.0f, 15.0f, 1.0f }, 0);
```

This light can be toggled by pressing the `0` button on the keyboard.

`GL_LIGHT1` is a specular light with the position `(20, 40, -40)` and the color `(1,1,1)` as can be seen in the following line.

```
// Configure light 1
Gdx.gl11.glLightfv(GL11.GL_LIGHT1, GL11.GL_SPECULAR, new float
[] { 1f, 1f, 1f, 1.0f }, 0);
Gdx.gl11.glLightfv(GL11.GL_LIGHT1, GL11.GL_POSITION, new float
[] { 20.0f, 40.0f, -40.0f, 1.0f }, 0);
```

This light can be toggled by pressing the `1` button on the keyboard.

`GL_LIGHT3` is then a gray ambient light and cannot be toggled.

We then have a diffuse value, specular value and shininess value for the Sphere. You should see it when you turn on and off the lights how these lights affect the material of our Sphere. Try it out and move the lights around at your own will and changes their colors.

Keys

In this example we let our ApplicationListener implement an interface named InputProcessor and we set our ApplicationListener to use it own instance to control inputs. We do this as follows.

```
public class First3D_Core implements ApplicationListener, Inpu
tProcessor {

    public void create() {
        // This class implements input processor that we use
        // within this application listener.
        Gdx.input.setInputProcessor(this);
    …
```

This class gives us more fine grained functions to deal with user input. You can use this interface to, for instance, detect when keys go up and down not just when they are pressed.

As stated above we have the numbers to control the lights we also bound the `q` button to exit the application. This is handy when you are running in game mode and you want to exit the application. You can stop a running LibGDX application by calling the exit function, that is `Gdx.app.exit();`.