Next, replace the GL_REPEAT parameter in the

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
```

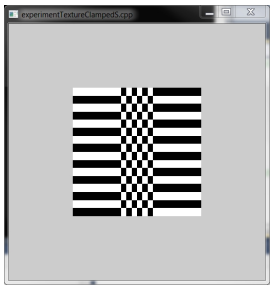

Figure 12.13: Screenshot from Experiment 12.8.

statement of both the loadExternalTextures() and loadProceduralTextures() routines with GL_CLAMP so that it becomes

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
```

This causes the wrapping mode to be clamped in the s-direction. It's probably easiest to understand what happens in this mode by observing in particular the chessboard texture: see Figure 12.13. Texture s coordinates greater than 1 are clamped to 1, those less than 0 to 0. Precisely, instead of the texture space being tiled with the texture, points with coordinates (s, t), where $s > 1$, obtain their color values from the point $(1, t)$, while those with coordinates (s, t), where $s < 0$, obtain them from $(0, t)$. End

Experiment 12.9. Continue the previous experiment by clamping the texture along the t-direction as well. In particular, replace the GL_REPEAT parameter in the

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

statement with GL_CLAMP. We leave the reader to parse the output. End

The repeating option is appropriate to tile the surface of an object with a particular pattern, e.g., a wall with a brick pattern, a table with a wood grain pattern, the ground with a grass pattern and so on, while the clamping option is appropriate to paint on a single copy of the texture, e.g., the facade of a building onto a rectangle, with the texture boundary, typically, aligned with the rectangle boundary.

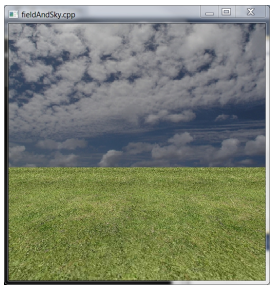

Figure 12.14: Screenshot of fieldAndSky.cpp.

12.3 Filtering

Experiment 12.10. Run fieldAndSky.cpp, where a grass texture is tiled over a horizontal rectangle and a sky texture clamped to a vertical rectangle. There is the added functionality of being able to transport the camera over the field by pressing the up and down arrow keys. Figure 12.14 shows a screenshot.

As the camera travels, the grass seems to *shimmer* – *flash* and *scintillate* are terms also used to describe this phenomenon. This is our first encounter with the *aliasing* problem in texturing. Any visual artifact which arises owing to the finite resolution of the display device and the correspondingly "large" size of individual pixels – at least to the extent that they are discernible to the human eye – is said to be caused by aliasing. End