Reflections on the SAGE Reflection Bug

Ian Parberry
April 16, 2009

The reflection of the terrain in SAGE seems to be not working under Windows Vista. At first I thought that it was only NVidia 8800 video cards and above, but that turned out to be a red herring. Here’s a screenshot taken under Vista in which you can see, or rather can’t see the problem. No reflections.

Yet under XP running exactly the same executable on exactly the same computer (it’s dual boot Vista/XP) we do get proper terrain reflection, as you can see below.
It seems to be the part of the code that puts in a clip plane at the water level so that only the terrain above the water gets reflected. That happens in the following line of code underlined in red in Reflection::beginReflectedScene.

```
void Reflection::beginReflectedScene(const Plane &plane) {
    // save current camera position and orientation
    m_oldPosition = gRenderer.getCameraPos();
    m_oldOrientation = gRenderer.getCameraOrient();
    // reflect the orientation and the position over the reflection plane
    EulerAngles newOrientation = plane.reflectOrientation(m_oldOrientation);
    Vector3 newPosition = plane.reflectPoint(m_oldPosition);
    // set the new position and orientation
    gRenderer.setCamera(newPosition, newOrientation);
    // setup the clipping plane
    Plane clipPlane = plane;
    // LOOK HERE
    // this is apparently what's giving the trouble on Nvidia 8800s. Terrain is not reflected.
    // Comment out this line and all is good except that what's underwater is reflected too
    gRenderer.setClipPlane(clipPlane);
    // clear screen for rendering
    gRenderer.clear(kClearFrameBuffer | kClearDepthBuffer | kClearToFogColor);
    gRenderer.beginScene();
    return;
}
```

If we comment out that line of code we get reflections back on Vista:
But then we get some nasty artifacts due to the fact that we’re reflecting underwater things that we shouldn’t be, so it’s not a good fix. The next screenshot shows those artifacts in the form of long, skinny triangles that I’ve indicated with arrows.

Looking into a little deeper, the call to `Renderer::setClipPlane` that was underlined in red above executes the following fairly innocuous code:

```cpp
2201 // sets a clipping plane
2202 /// \param plane array of 4 floats that represent a plane
2203 void Renderer::setClipPlane(const Plane &plane)
2204 {
2205     pD3DDevice->SetRenderState(D3DRS_CLIPPING, true);
2206     pD3DDevice->SetRenderState(D3DRS_CLIPPLANEENABLE, D3DCLIPPLANE0);
2207     clipPlane = plane;
2208     pD3DDevice->SetClipPlane(0, clipPlane);
2209     clipPlaneEnable = true;
2210 }
```

All three of those DirectX calls (two to `SetRenderState` and one to `SetClipPlane`) return D3D_OK under Windows Vista as expected. Changing the clip plane index from 0 to 1 or 2 doesn’t help. Variable `clipPlane` does contain the expected values when line 2207 is executed. There’s nothing that indicates that there’s a problem here, unfortunately.

The solution? Don’t use Windows Vista I suppose.