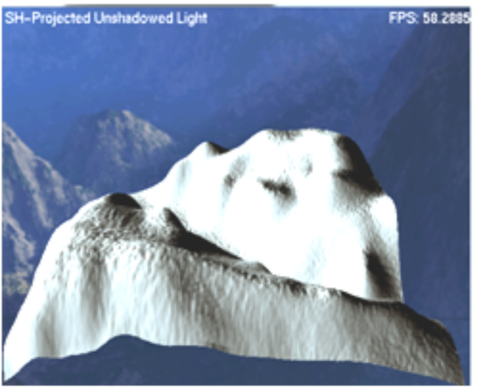
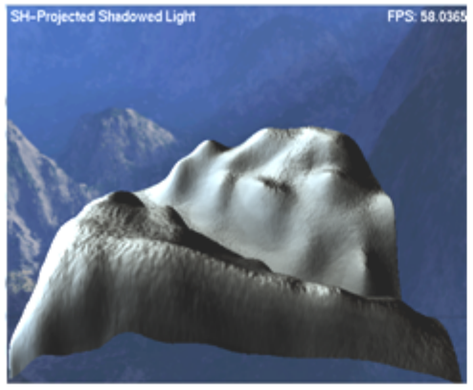
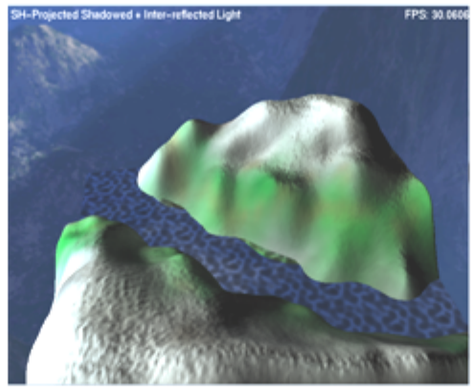
-RESULTS-








```
=========================================
KEYS:
Q/E : Toggle application modes
W/S/A/D : Move camera (in certain appmodes)
R/F : Up/Down camera (in certain appmodes)
U : Toggle water surface On/Off (in certain appmodes)
Left click + Mouse : Rotate View
=========================================

******LOADING******
Number of SH samples : 300
Preparing spherical function stratified samples...          [Done]
Loading time : 0.000000 seconds

Projecting test light function to SH coefficients...        [Done]
Loading time : 0.000000 seconds

Loading textures...                                         [Done]
Loading time : 0.624000 seconds

SH calculations for unshadowed SH lighting...               [Done]
Loading time : 0.437000 seconds

SH calculations for shadowed SH lighting...                 [Done]
Loading time : 2.294000 seconds

SH calculations for untextured shadowed indirect SH lighting... [Done]
Loading time : 7.519000 seconds

SH calculations for coloured shadowed indirect SH lighting...   [Done]
Loading time : 7.457000 seconds


Total loading time : 18.331 seconds
```

~60fps w/o water plane ~30fps with

-CONCLUSION-

✓**Successful, lightweight and compact implementation of a real-time Global Illumination algorithm with significantly low hardware requirements**

FURTHER WORK :

- **Optimise algorithm**
- **Dynamic objects**
- **True real-time**

Acknowledgements

I would offer my greatest thanks to the following people :
Matthew Bett – Technical Supervisor
Dr. Suheyl Ozveren – Module lecturer
My wife and family

-Algorithm pseudo code-

```
// Precomputation:
Prepare the stratified spherical samples;
Load the bitmaps;
Create 'mesh' object;

// Vertex basic initialization
foreach(vertex in mesh)
{
                initVertex(vertex, X,Y, height, colour, normal, etc...)
                vertexPointers[X][Y] = vertex;
}

// Vertex SH coefficients initialization
foreach(vertex in mesh)
{
                projectVertexConsideringSelfShadowing( vertex );
}

// Inter-reflected light calculation - additional step after shadowing
accumulatedLightSHcoeffs[num_of_vertices];
for(number_of_interreflected_bounces) // usually one is enough
{
                foreach(vertex in mesh)
                {
                                accumulatedLightSHcoeffs[vertex] =
                calcIndirectCoef( vertex );
                }

                foreach(vertex in mesh)
                {
                                vertex.SHcoeffs +=
                accumulatedLightSHcoeffs[vertex];
                }
}

// OpenGL display method called each frame, real-time
display()
{
  Rotate the hardcoded HDR environment light coefficients;

  foreach(vertex in mesh)
  {
    vertex.finalColour = dotProduct(vertex.SHcoeefs, rotatedLightCoeffs);
  }

  Draw the vertices using their finalColour;
}
```