Polygon Shadow Generation

We used the algorithm in the paper *POLYGON SHADOW GENERATION* by Peter Atherton et al.

The algorithm works by comparing regions that the user can see from the sun view to the regions user can see from the user view. The region that is not present in the sun view but is visible to the user is made darker compared to its surroundings making the region part of the shadow.

This algorithm uses hidden surface removal which we implemented from the paper *HIDDEN SURFACE REMOVAL USING POLYGON AREA SORTING*by Kevin Weiler andPeter Atherton.

Hidden surface removal uses the Sutherland Hodgman polygon clipping algorithm in its core.

**Implementation -**

1. Give inputs for the surfaces of the polygons, location(s) of the light source(s), location of the user, and location of the base plane.
2. This configuration is then copied and saved for later use. The entire scene is then translated into sun view (both original and copy version).
3. A depth sort is performed on the given polygons (and copied polygons).
4. Starting from the farthest polygon from the sun, the last entry of the depth sorted array, hidden surface removal is performed for every pair of polygons. New coordinates are stored in an array.
5. Both the copied version of the scene as well as the modified version of the scene are transformed into the user view.
6. Both the scenes are compared and the regions which are not common are made a little darker.
7. The final version is rendered on the screen.

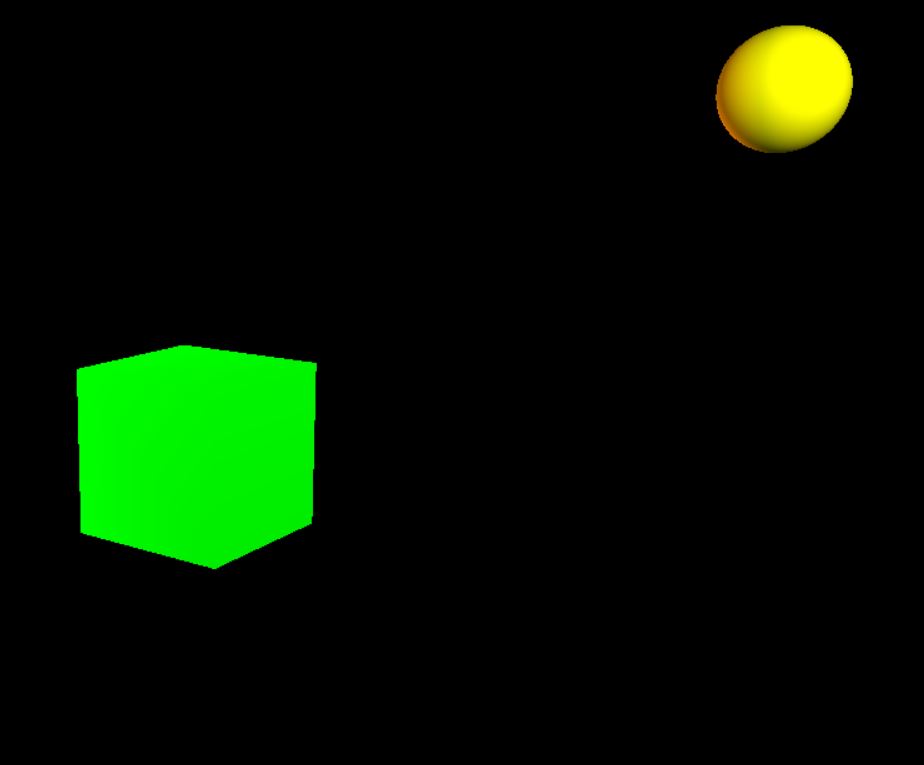
**Results –**

NOTE: Hidden Surfaces are coloured Blue while visible Surfaces are coloured Red. Two white coloured screens are used to observe the shadows created by the objects in the scene.

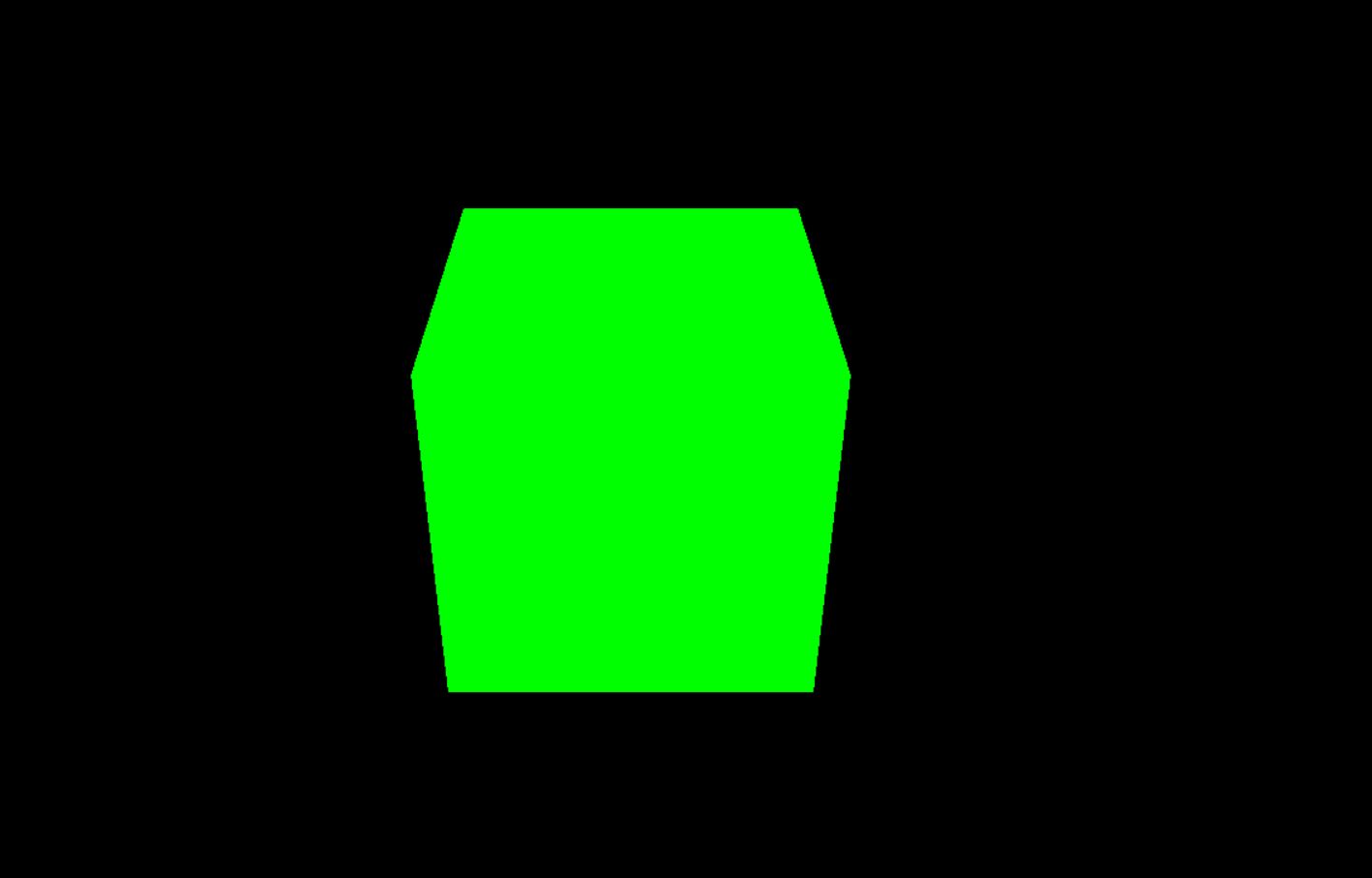
**TEST CASE 1 : SIMPLE CUBE**

This test case is used as very fundamental check for the code, the lighting used in diffused so as to provide better idea of the distance and direction from the light source.

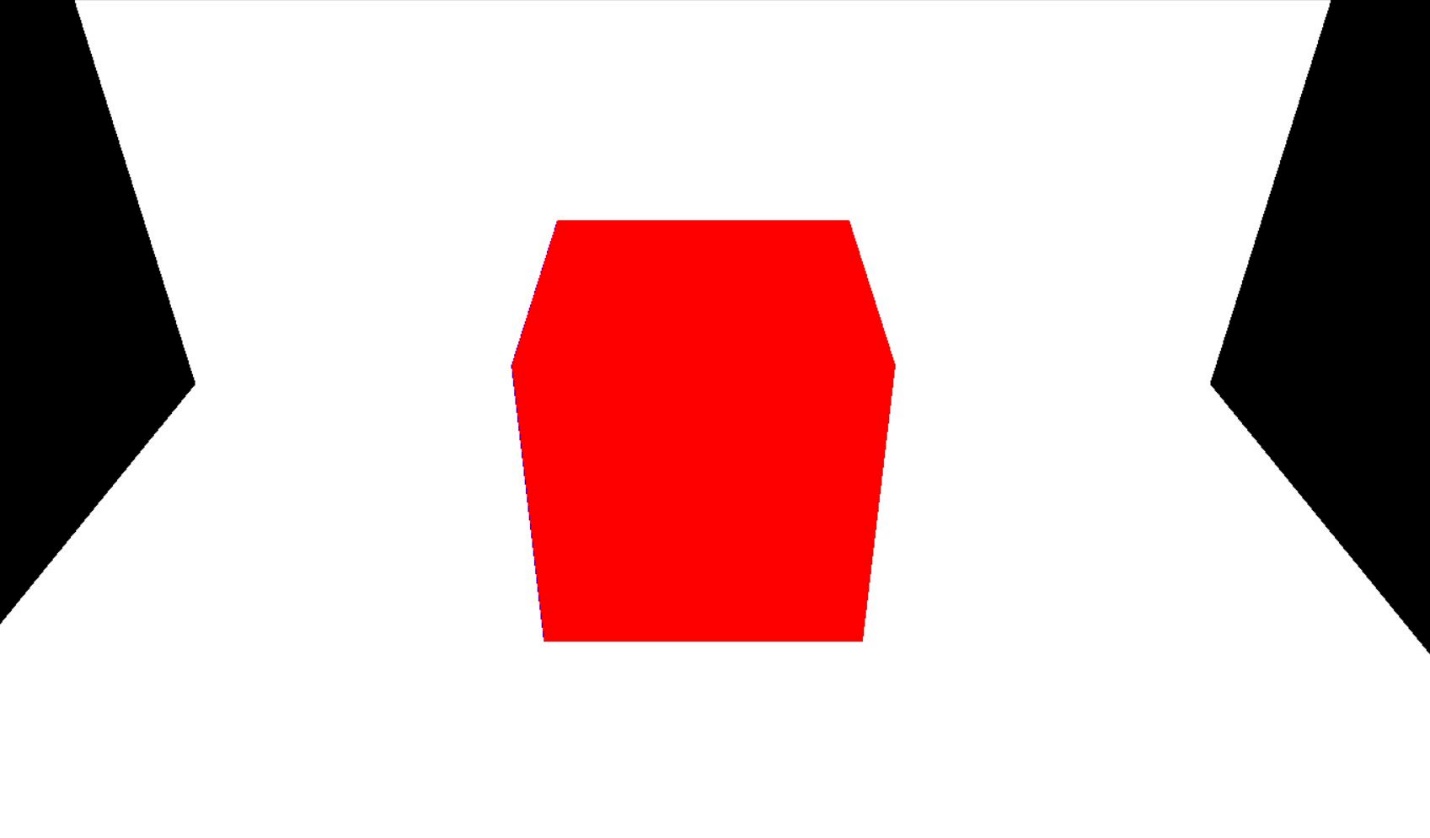
The test case checks if an object is being correctly clipped and shadow is being generated clearly or not.



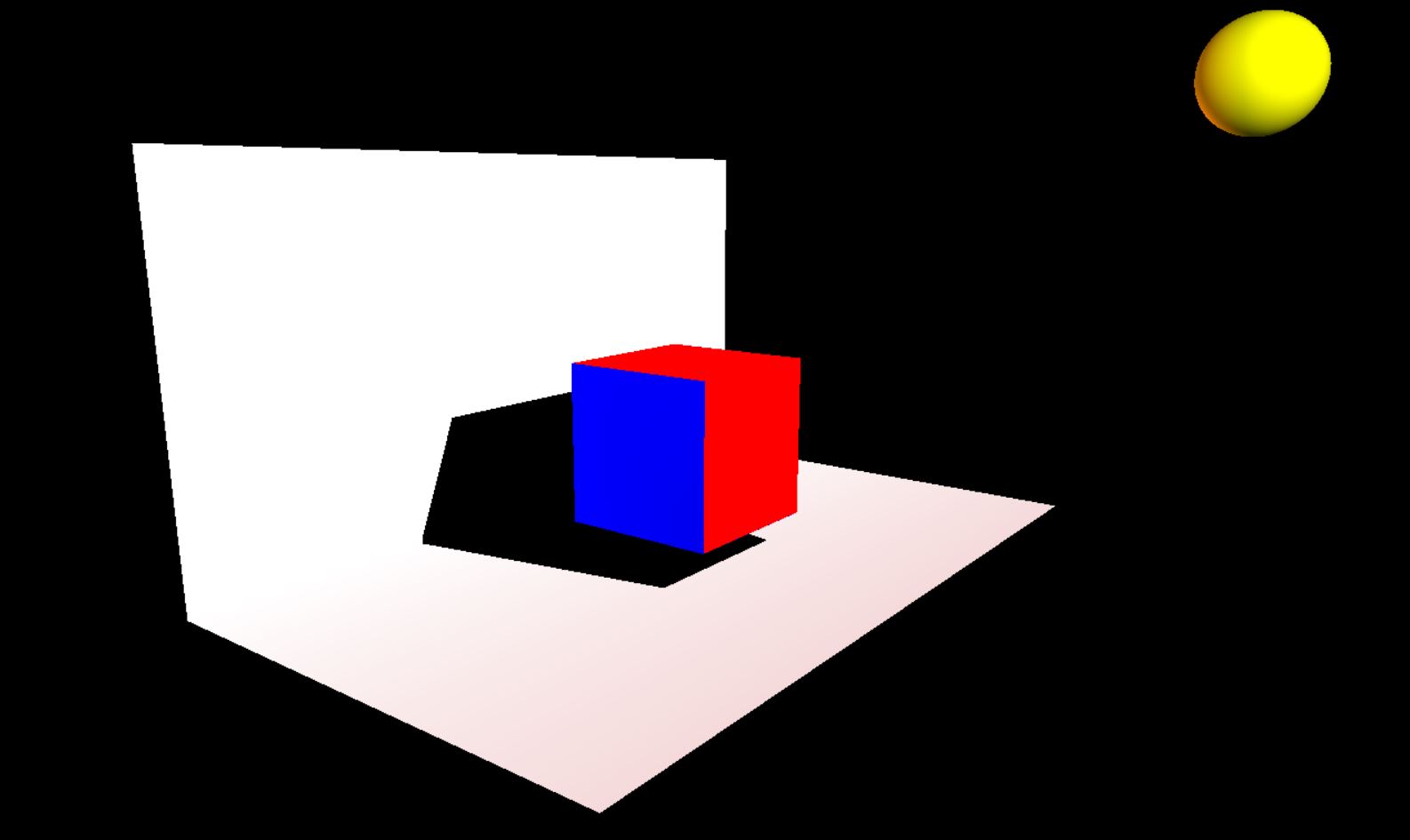
This is the side view for the cube which is to be clipped. The sun represents our light source for the scene.



View from the Sun, only the top and front faces are visible, rest all should be hidden.



View from the Sun, after clipping is done, the visible faces are coloured red.



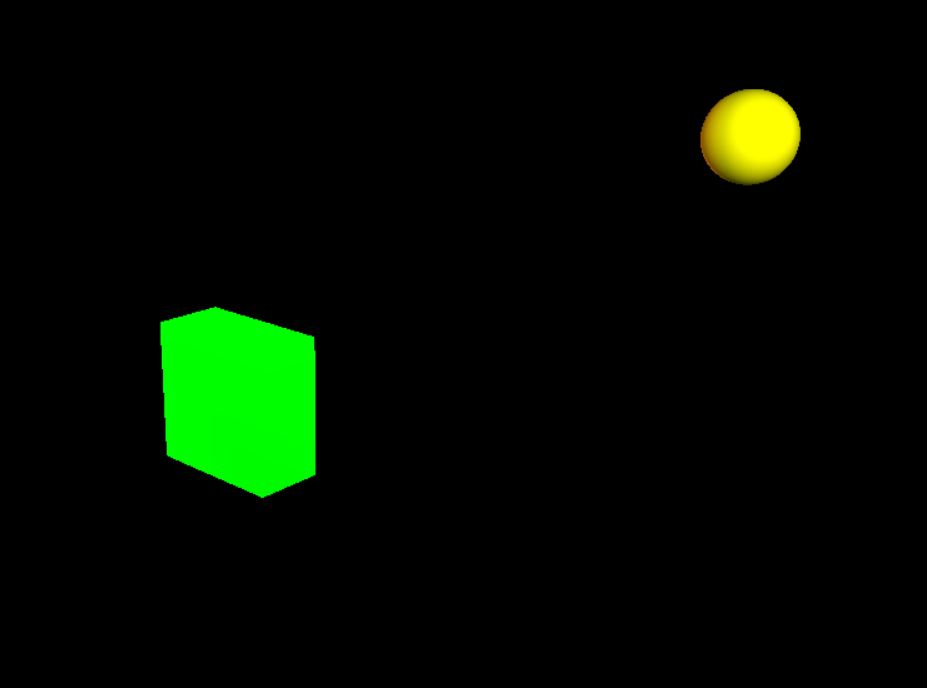
Side View of the final clipped scene. Blue coloured surface is not visible from Sun’s view. Black shadow is formed on the planes, corresponding to the objects obstructing the light.

This is the final image of the simple cube scene.

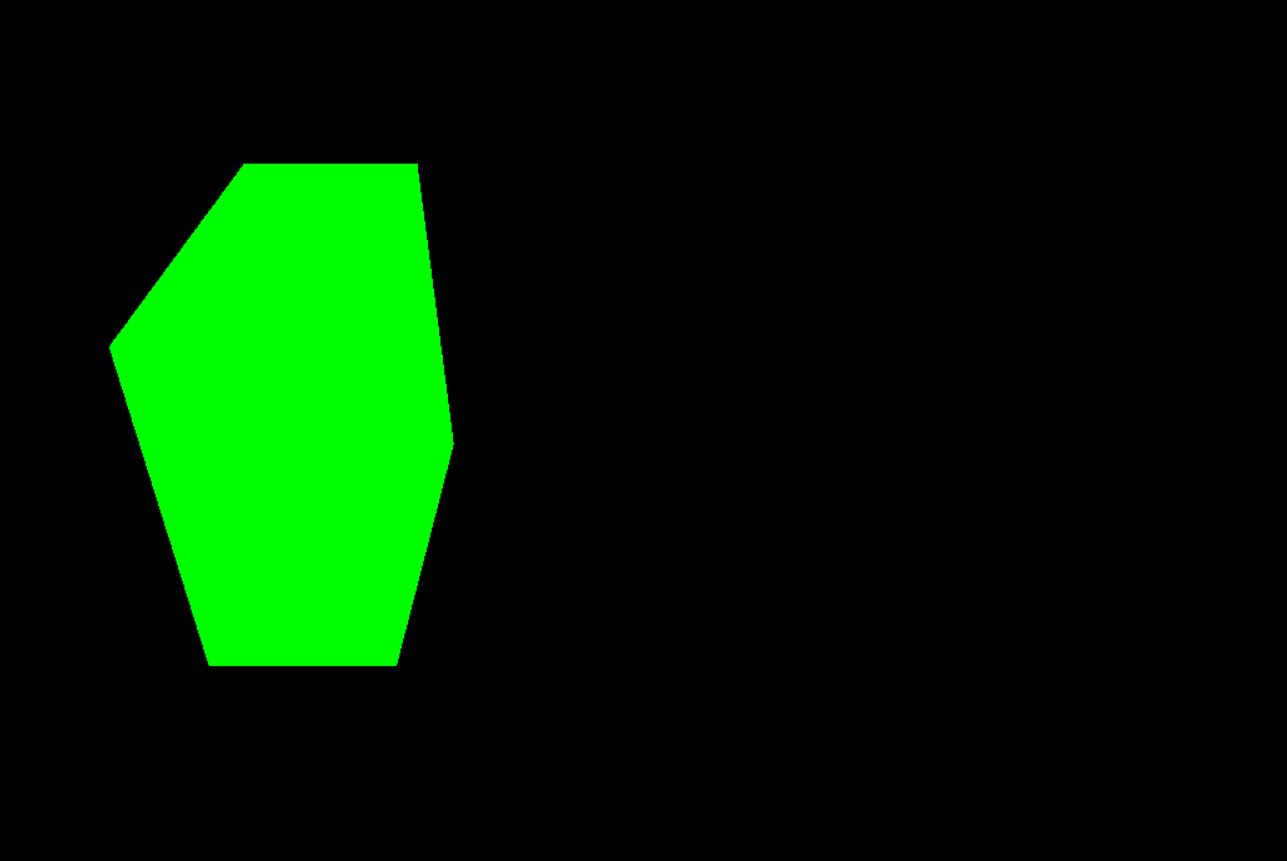
**TEST CASE 2 : SIMPLE CUBOID**

This test case checks if an object which is not placed directly in front of the Sun, but rather at some angle from it, is being clipped correctly or not.

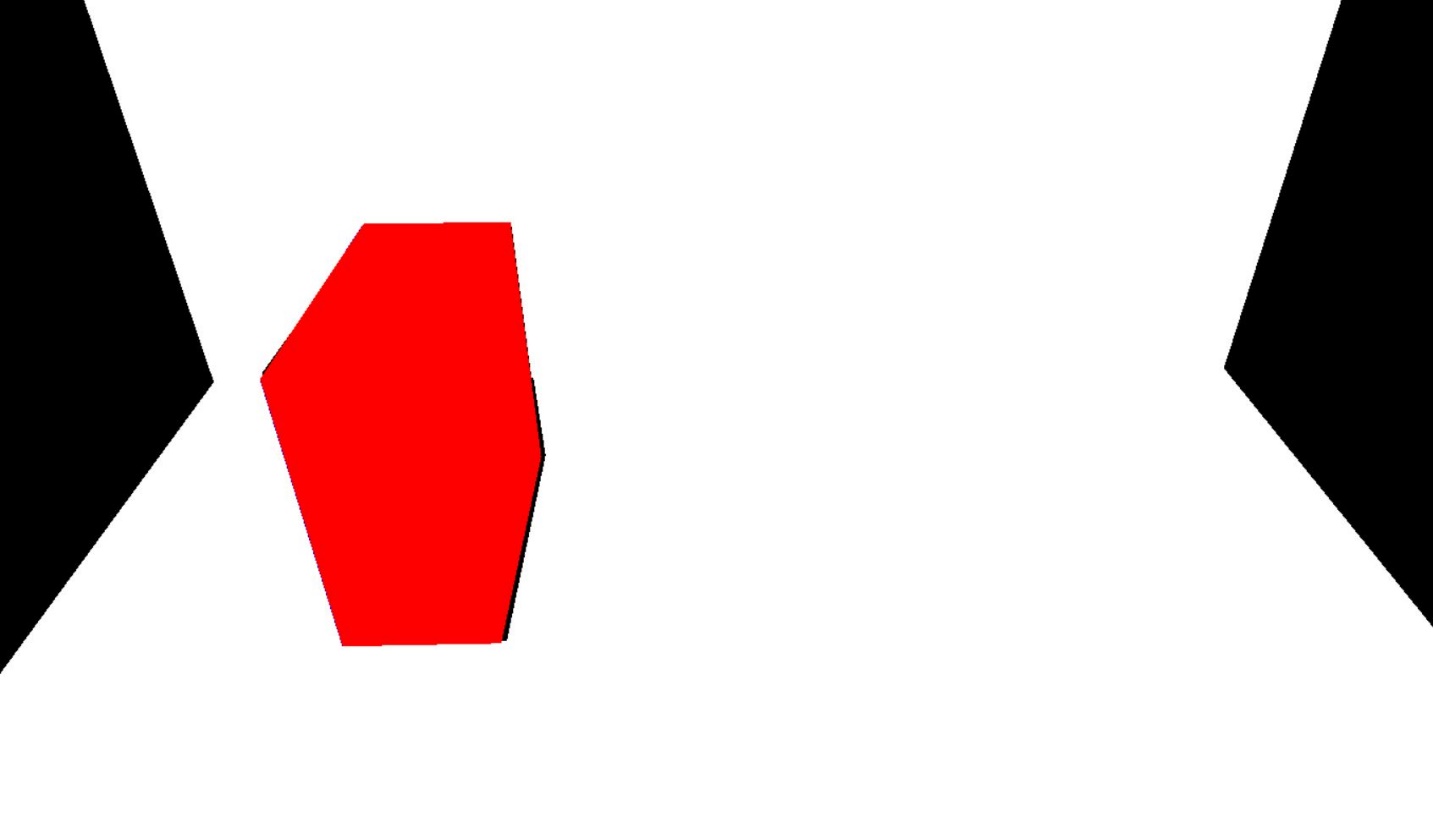
The cuboid is placed at some angle from the Sun to test the code’s ability to clip angled objects as well.



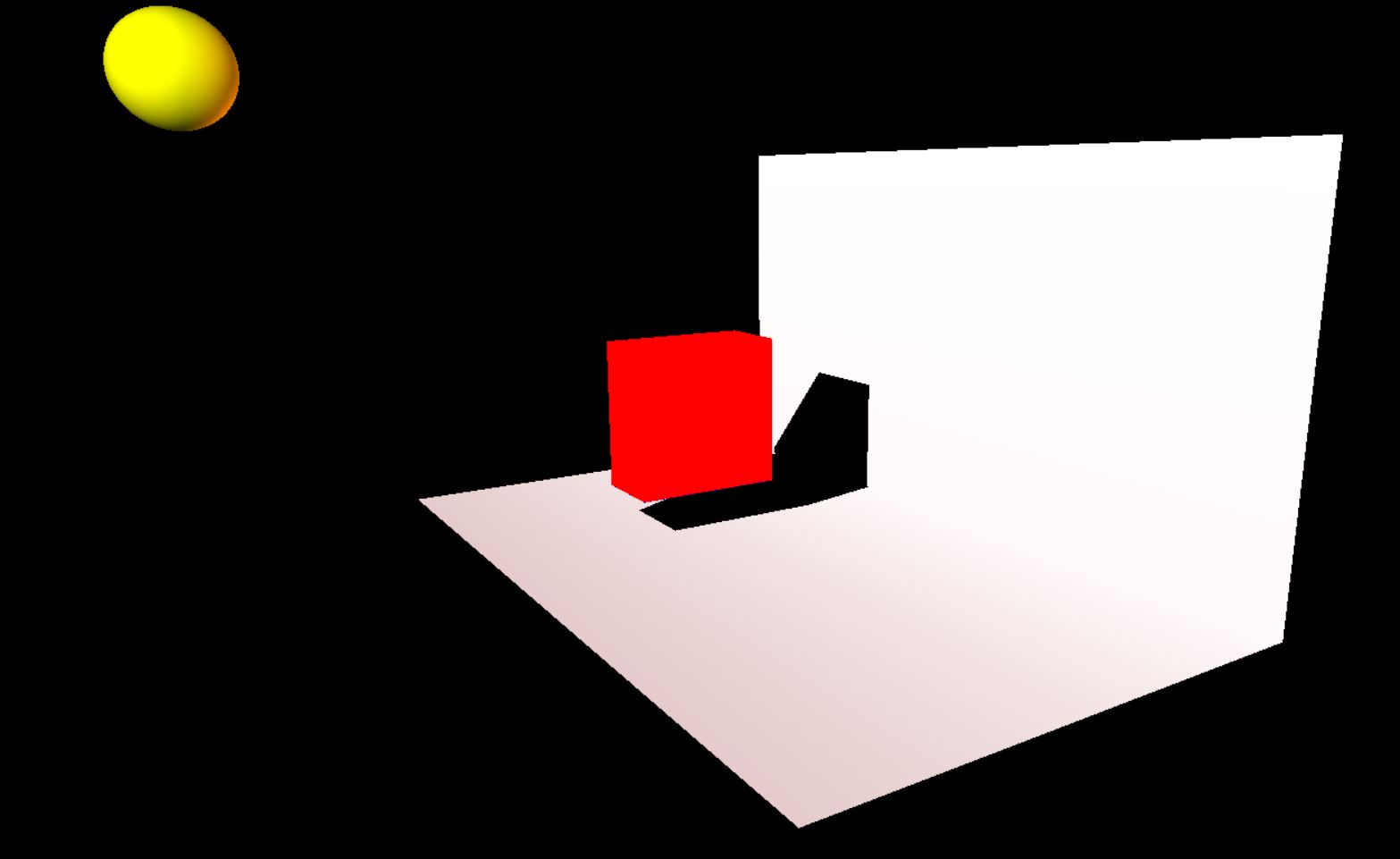
This is a side view of the cuboid in sun case. The light source is placed at Sun’s coordinates.



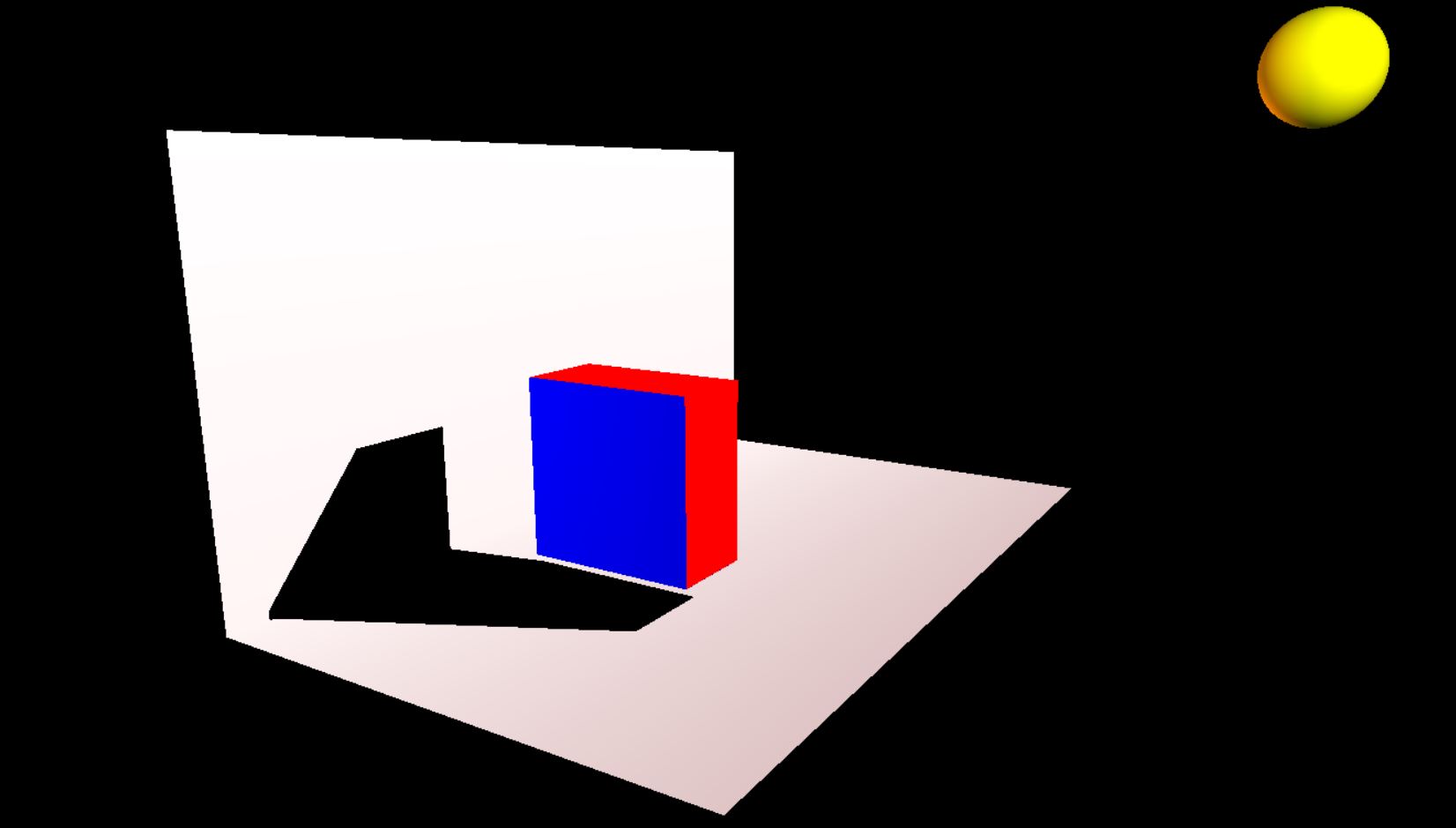
View from the Sun, only the top, front and right faces of the cuboid are visible.



View from the Sun, the front, top and right are coloured red as they are visible, others are coloured blue as they are hidden surfaces.



Side View of the cuboid from the angle where right face is visible to the viewer, it can be observed that the shadow of the object is falling on the screens placed.



The clipped polygon from the opposite angle than above, the left face is towards the viewer but away from the sun, hence it is not visible in the Sun view and is coloured Blue.

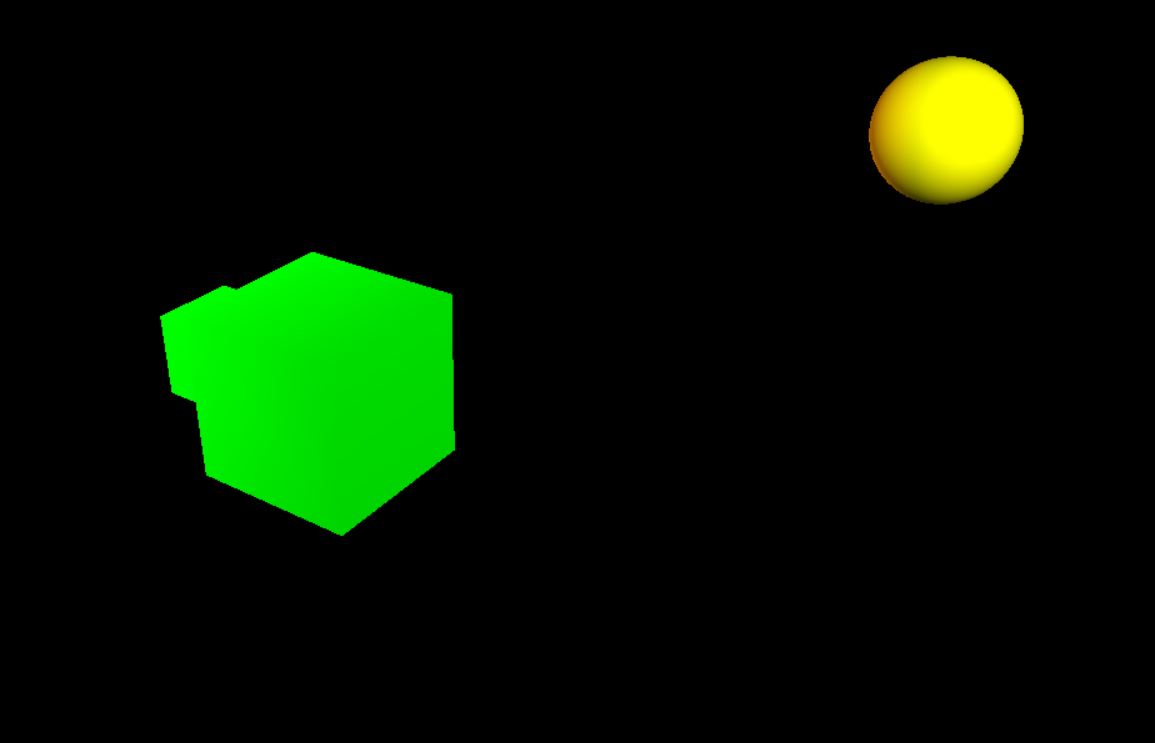
The shadows cast by the cuboid are visible on the screens.

Note: the shadows might appear skewed from the user’s view due to relative proximity of the Sun and the screens to the object.

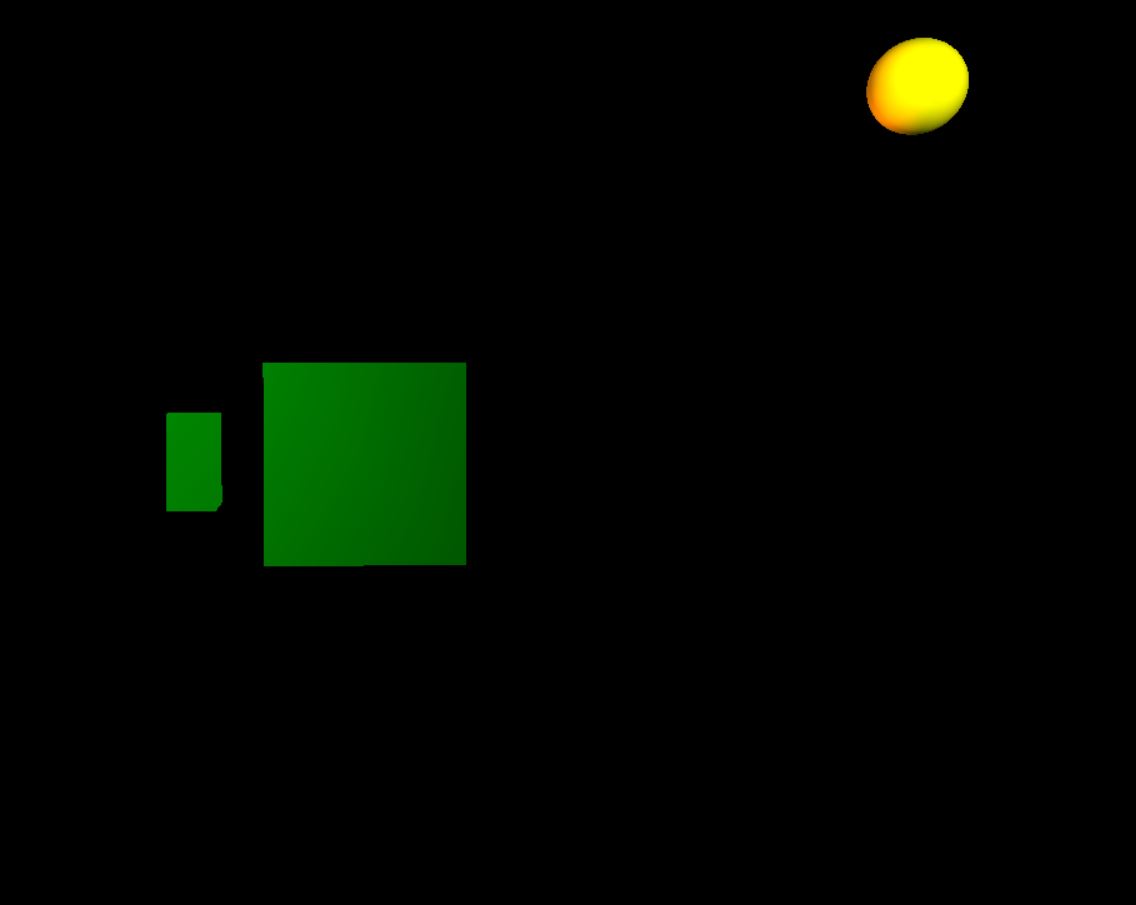
This is the final image for this test case.

**TEST CASE 3 : SIMPLE CUBE WITH AN OBJECT HIDDEN BEHIND IT**

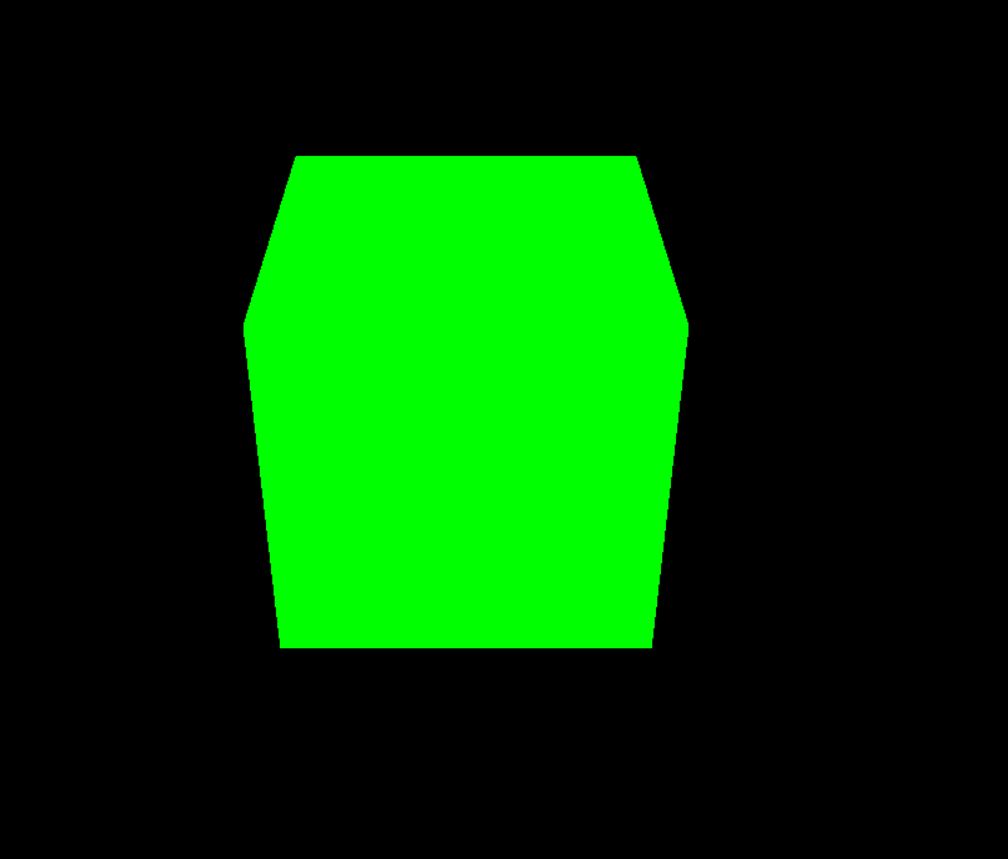
This test case checks if an object that lies entirely within the shadow of another object, being regarded as hidden as a whole or not. Here a small cuboid is placed behind a large cube and before the screen, this cuboid will not be visible from the Sun’s view and hence will be entirely Blue.



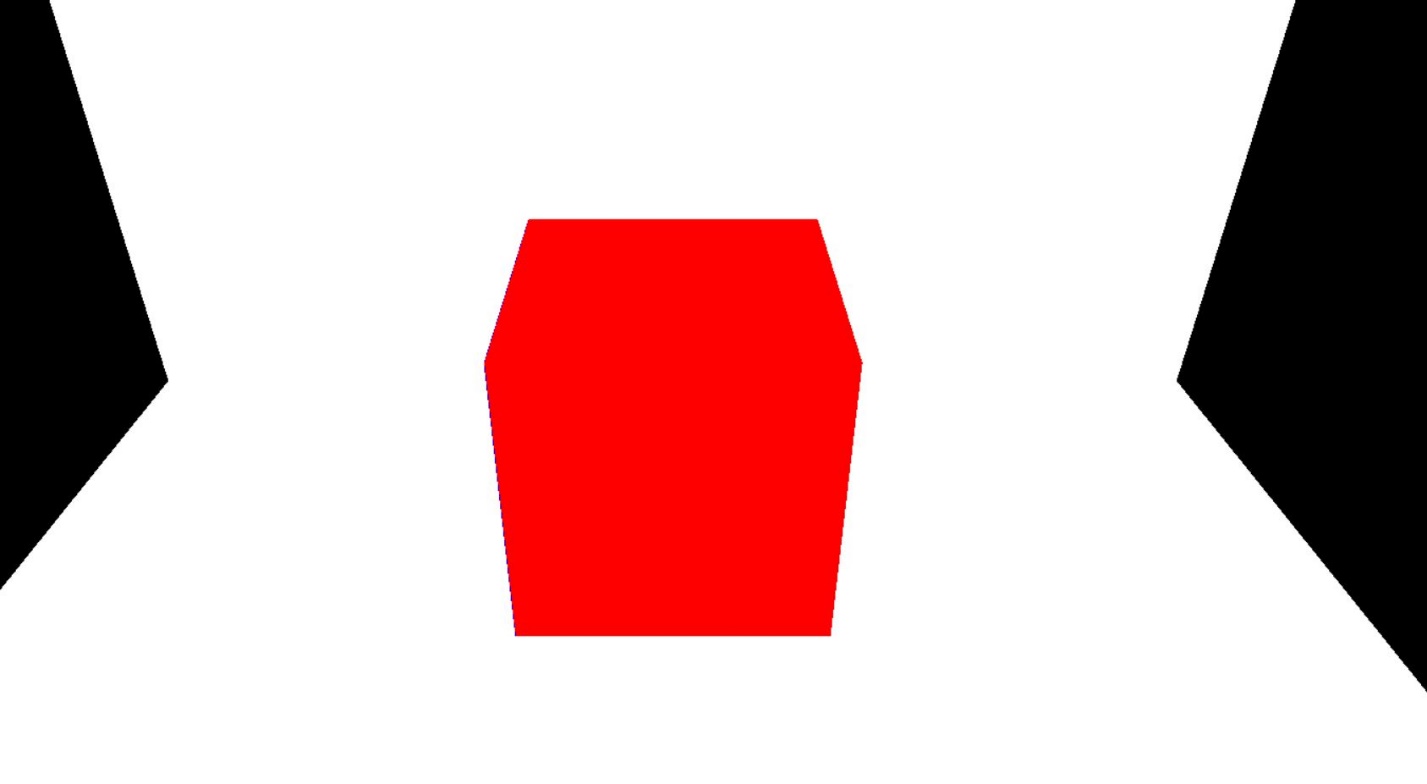
Angled Side View that shows an object placed behind the cube and far from the Sun .



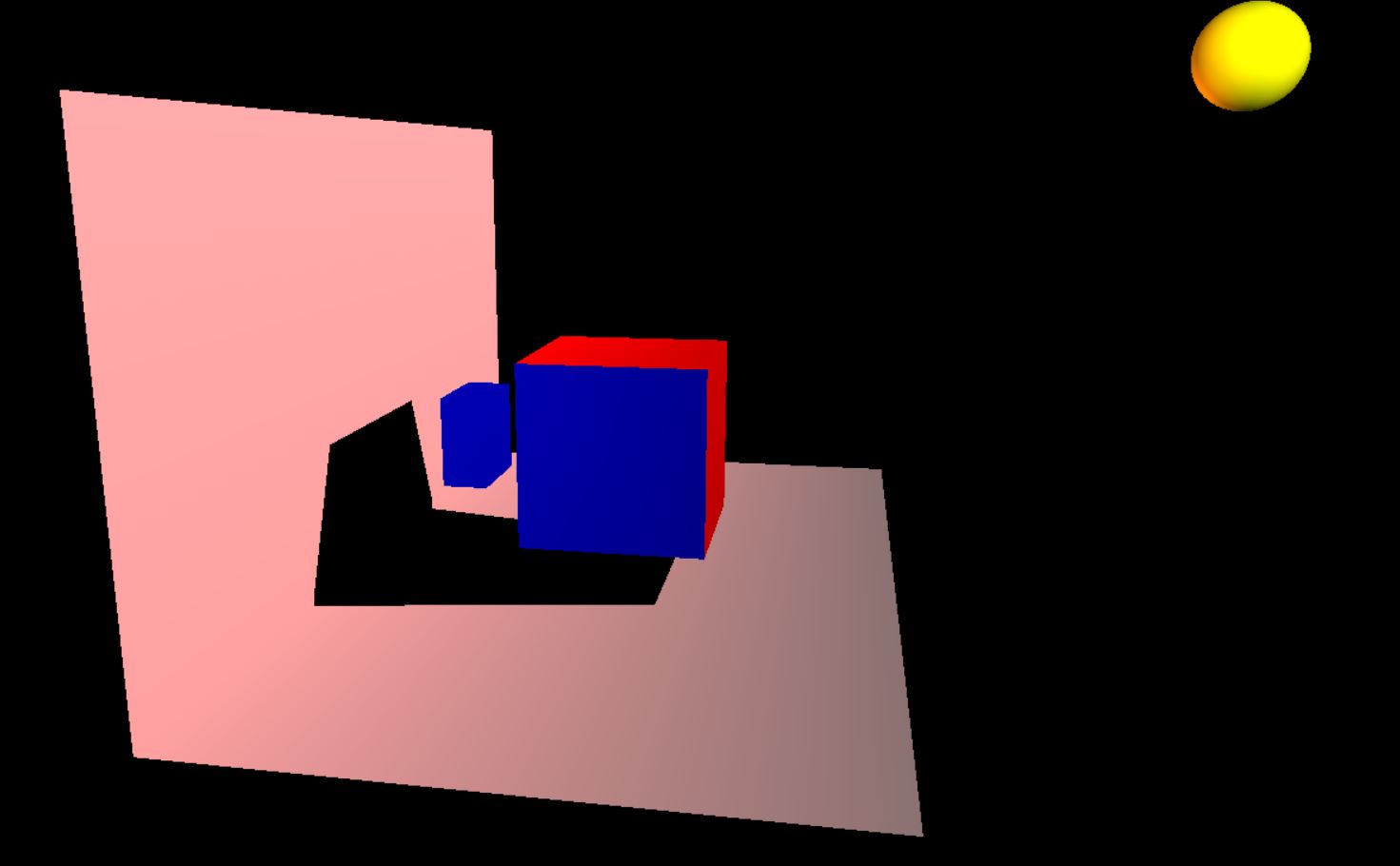
This is a Side View of the non-clipped scene that shows a separation between the small cuboid and large cube.



View from the Sun of the objects, it is observed that the cuboid is entirely behind the cube and hence only the top and front faces of the cube will be visible.



View from Sun, of the clipped scene, only the front and top surfaces of the cube are visible, the cuboid is entirely hidden.

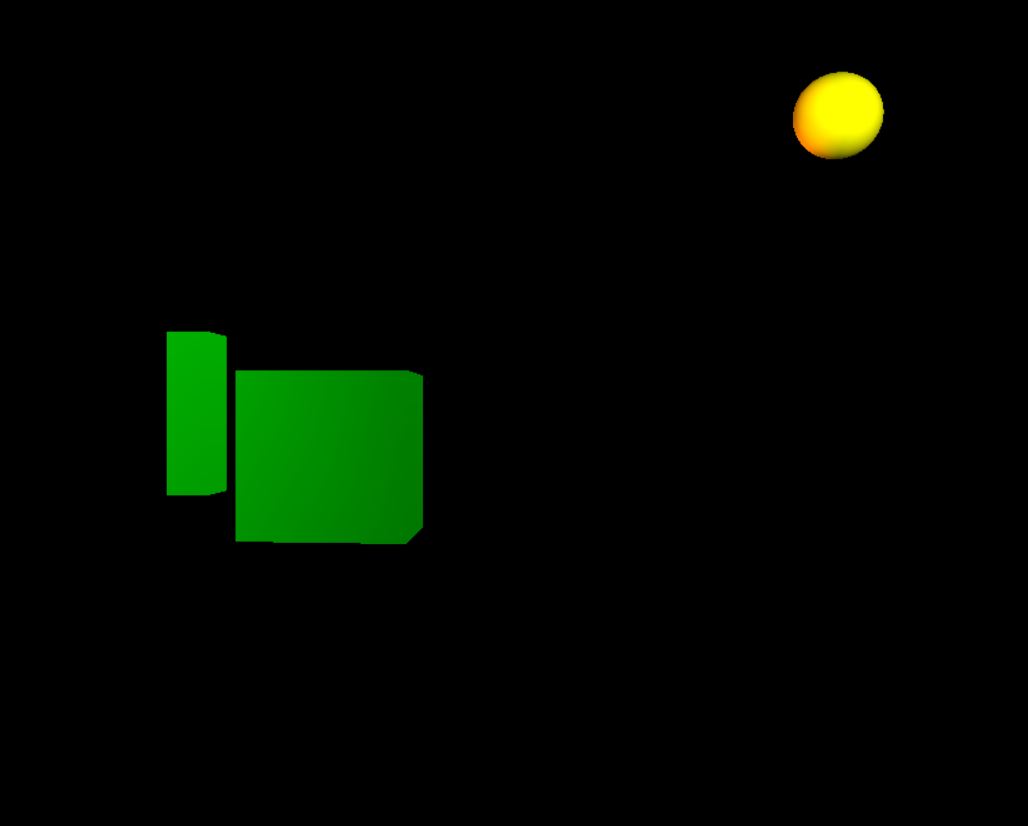


This is a Side View of the clipped scene that shows the entire cube in Blue (Hidden), the shadow of the cuboid is not visible on the screen because it does not obstruct any light from the light source. This is the final image for the test case.

**TEST CASE 4: SIMPLE CUBE WITH AN OBJECT PARTIALLY HIDDEN BEHIND IT**

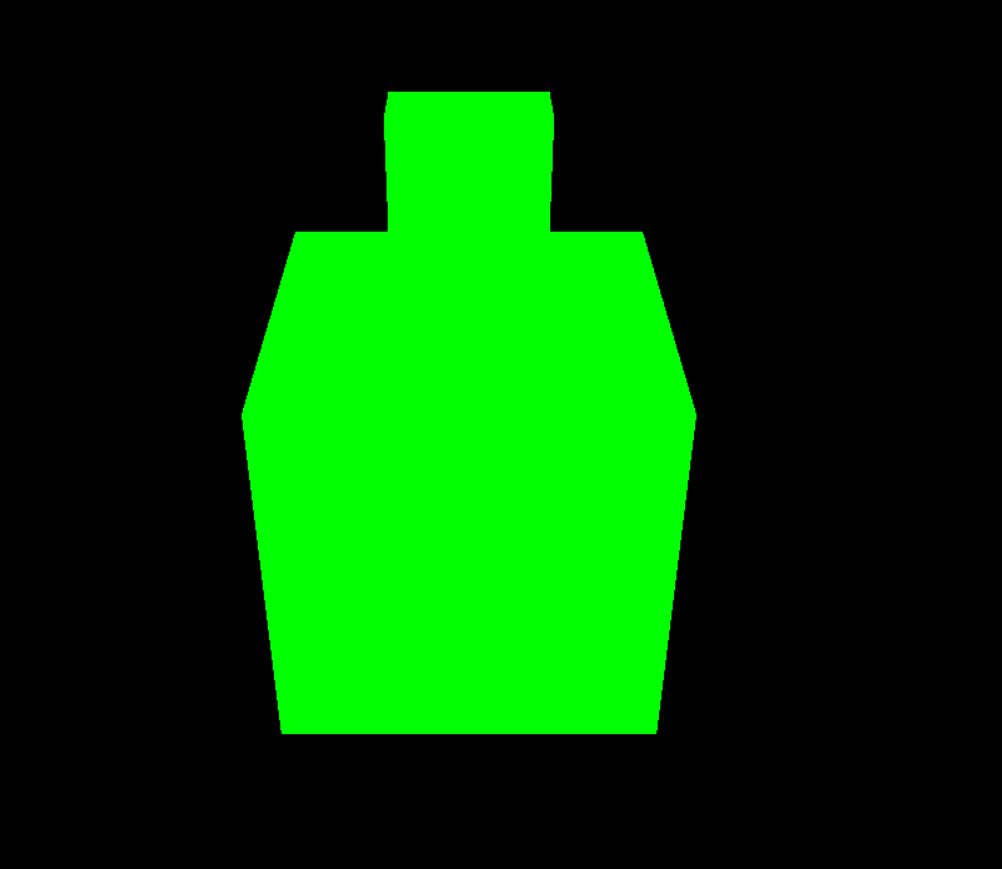
This test case aims to check that the code clips partially hidden objects or not.

A cuboid is placed behind a cube and far from the sun, such that part of it will be hidden behind the shadow of the cube.

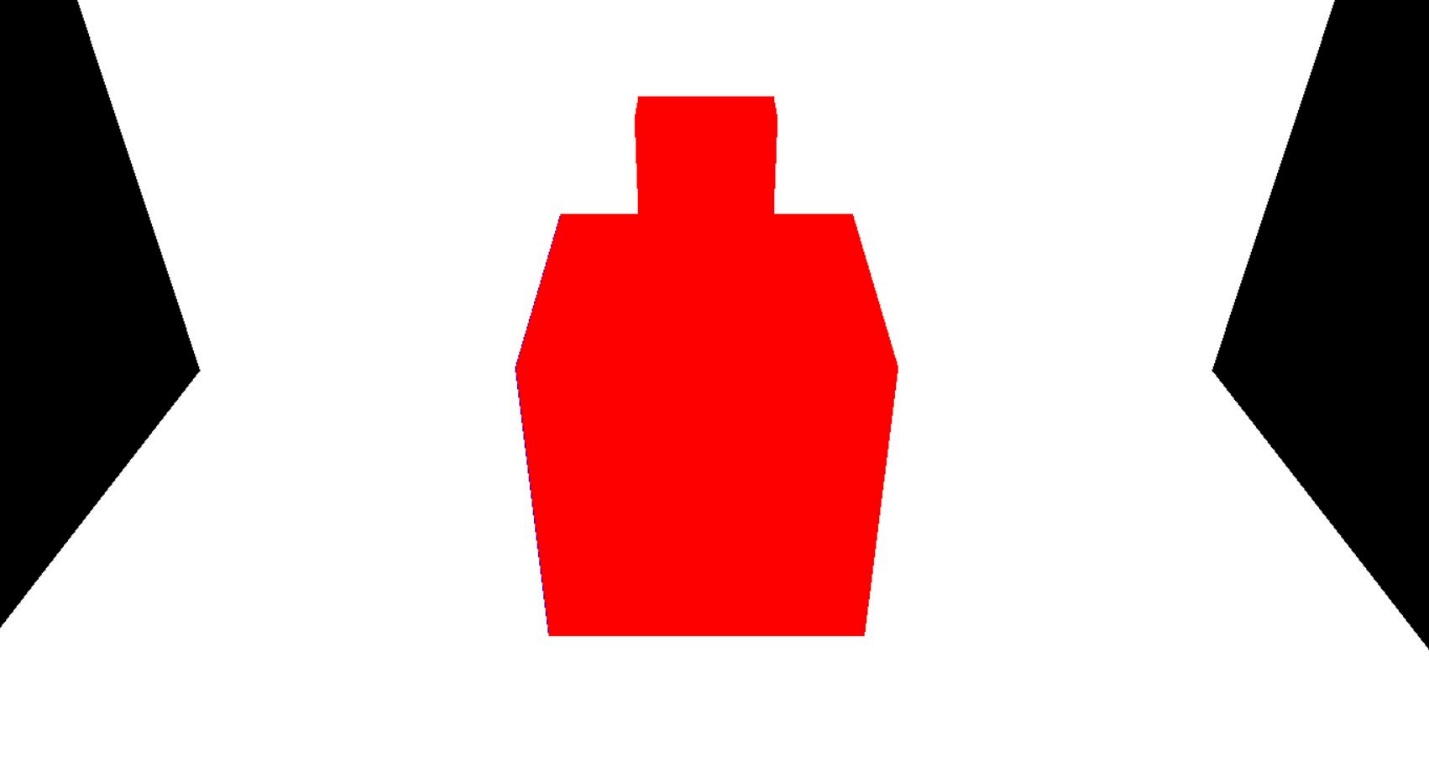


Side View that shows the separation between the objects and that some part of the cuboid is visible from the sun.

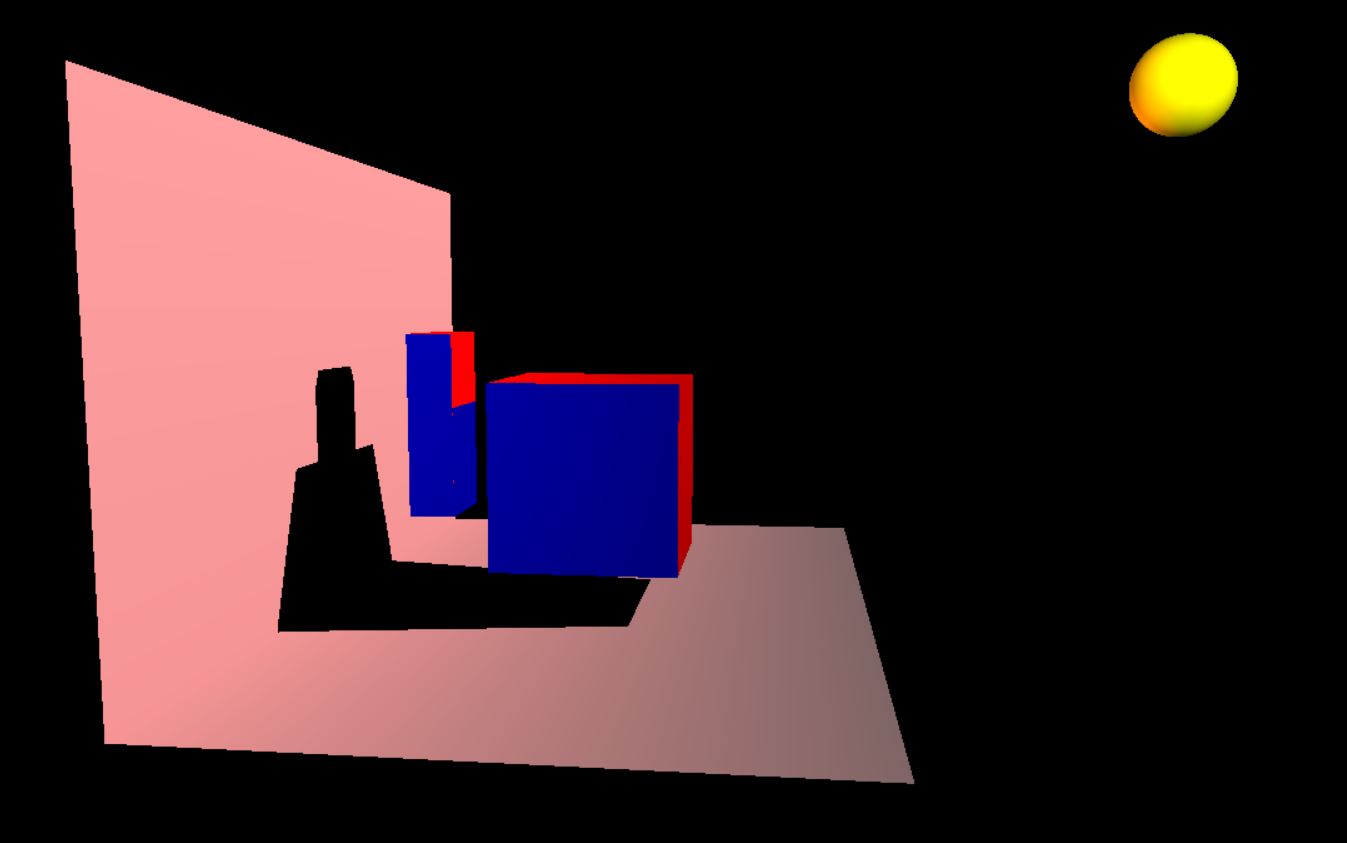
Note: the dull lighting is observed because the viewer is zoomed out to account for the relative distances between the objects and light source.



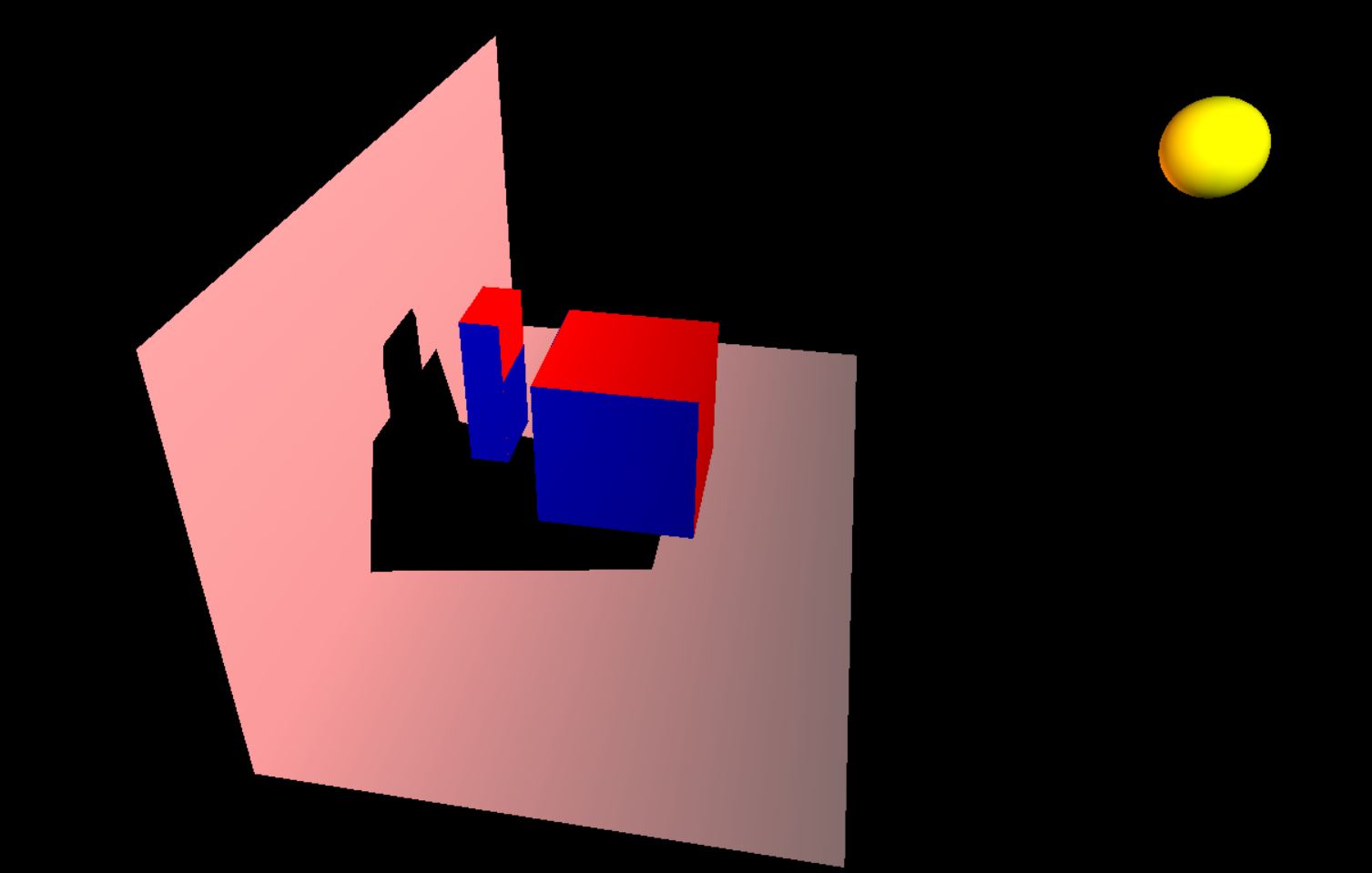
View from Sun, of the non-clipped scene, the front and top faces of the cube are visible. The top face of the cuboid and part of it’s front face is also visible.



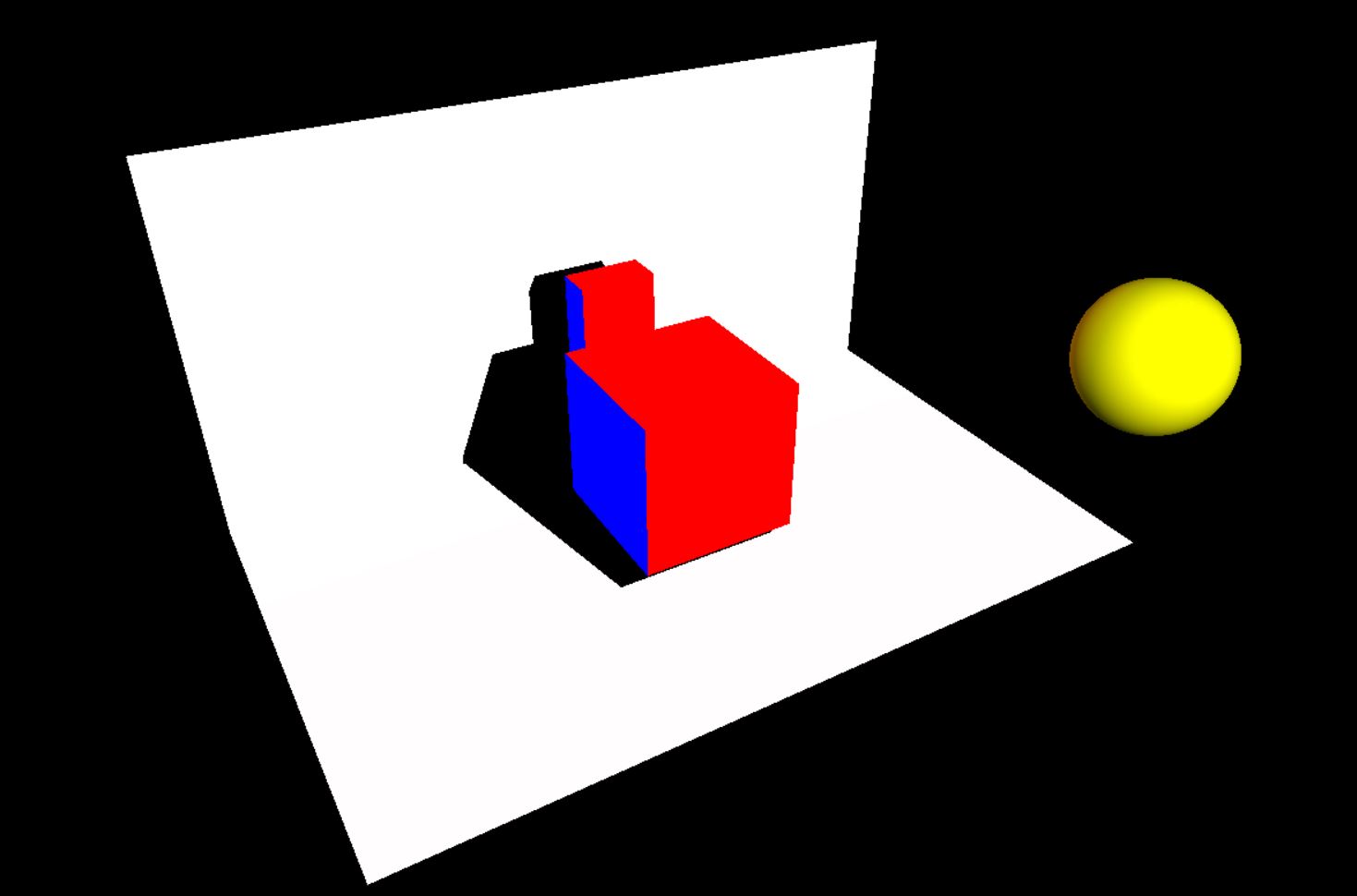
View from Sun, of the clipped scene, the front and top faces of the cube, along with the top face and part of front face of cuboid are coloured red, as they are visible from the Sun.



Side View of the clipped scene, the left faces of both cube and cuboid, don’t receive sunlight and are hence hidden and in Blue, the shadows of both cube and cuboid are visible of the screens.



This view shows that the top faces of both the polygons are coloured and their shadows are also being formed on the screens.



Angled View of the clipped scene, which shows some of the hidden surfaces and the non-hidden ones, it must be noted that the skewed shadow of the scene is a result of the L shaped screen.

This is the final image of this test case.

Hence it has been verified that the code works for all the above cases and that it provides the intended implementation of the Polygon Shadow Generation Research Paper, for polygons, irrespective of whether they are visible directly from the light source ( whether completely, partially or not visible at all) or if they are placed at an angle from the light source.

The code can be accessed by executing the “shadow2.c” file .