Test Report: Lights, Camera, Models!

Sasha Soraine

December 19, 2019

1 Revision History

Date	Version	Notes
December 18, 2019	1.0	Original Version.

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document is the System and Unit VnV Test Report for Lights, Camera, Models! Due to limitations with time, and the extensive use of Unity's built-in modules which were assumed to function correctly, this report will predominantly feature the manual testing outlined in System VnV.

3 Functional Requirements Evaluation

[I am unsure how to write about this evaluation given that all of my tests are manually comparing scene renders. —SS]

3.0.1 Run-Time Tests

All of these test cases share the following properties:

Initial State: Lighting Model = Lambert
ks = 0.5
kd = 1
$\alpha = 1$
Colour = ()
Specular Colour = White
Light Type = Spotlight
Light Colour = White

This means the default scene should look like this:
Our finished default scene looks like this:

Lighting Model Changes

1. lightModel-ModLambert

   Control: Manual

   Input: Select Mod-Lambert lighting model from dropdown list.

   Output: Scene render with Mod-Lambert lit sphere.

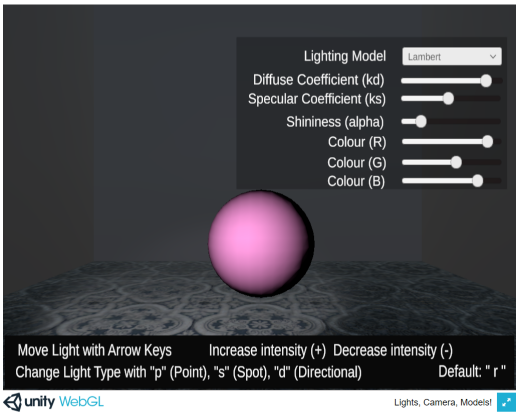

Figure 1: Expected: Default Scene

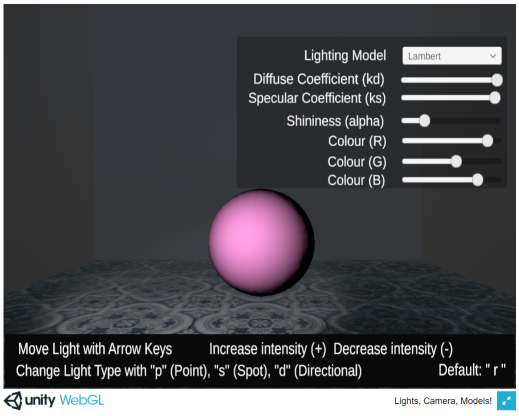

Figure 2: Actual: Default Scene for Lights, Camera, Models!

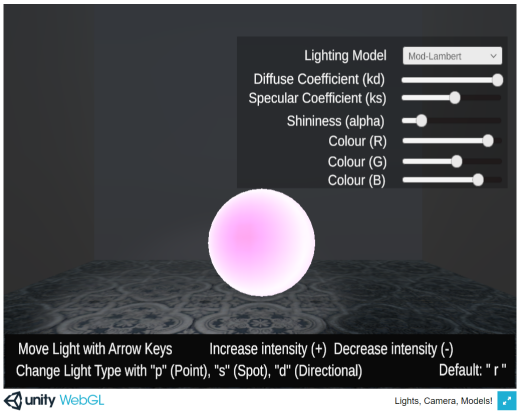

Figure 3: Expected Output: Default Scene with Lighting Model changed to Mod-Lambert

2. lightModel-Phong

   Control: Manual

   Input: Select Phong lighting model from dropdown list.

   Output: Scene render with Phong lit sphere.

   Test Case Derivation: After a scene is loaded, the user can only interact with the system through its GUI. Lighting model changes are determined by dropdown list selection. The scene default is to have a sphere lit with the Lambert (Diffuse) lighting model; changing to the Phong will make a significant difference and should be easily noticeable.

   How test will be performed: System will recalculate luminous intensity of points on objects in the scene using the new model. New luminous intensity information will be sent through the rendering pipeline to output visuals. Tester will visually compare whether output matches the image provided here.

3. lightModel-BlinnPhong

   Control: Manual

   Input: Select Blinn-Phong lighting model from dropdown list. [There isn't enough information here for someone else to do this test. The

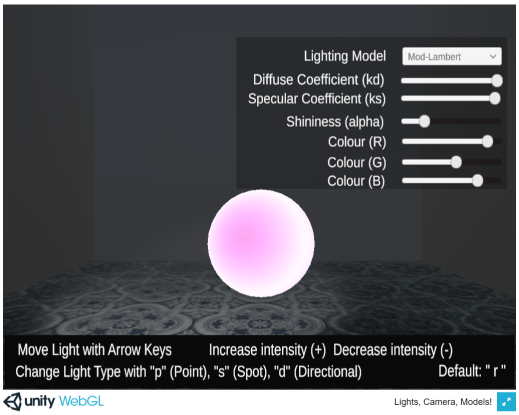

Figure 4: Actual: Default Scene for Lights, Camera, Models! with ModLambert

input is ambiguous. The same comment applies for other tests in this section. —SS]

Output: Scene render with Blinn-Phong lit sphere.

Test Case Derivation: After a scene is loaded, the user can only interact with the system through its GUI. Lighting model changes are determined by dropdown list selection. The scene default is to have a sphere lit with the Lambert (Diffuse) lighting model; changing to the Blinn-Phong will make a significant difference and should be easily noticeable.

How test will be performed: System will recalculate luminous intensity of points on objects in the scene using the new model. New luminous intensity information will be sent through the rendering pipeline to output visuals. Tester will visually compare whether output matches the image provided here.

Object Changes

1. objMaterialPropChange-valid-ks

   Control: Manual

   Initial State: Default Scene with lighting model set to Blinn-Phong (see 6)

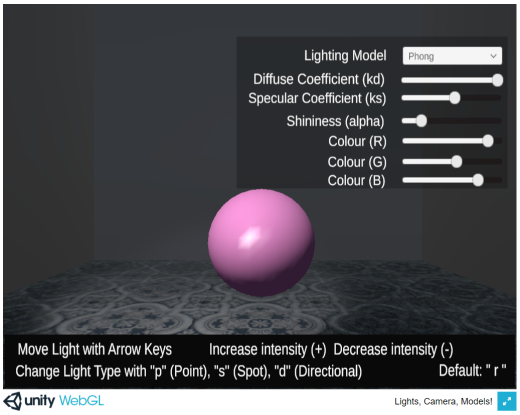

Figure 5: Default Scene with Lighting Model changed to Phong

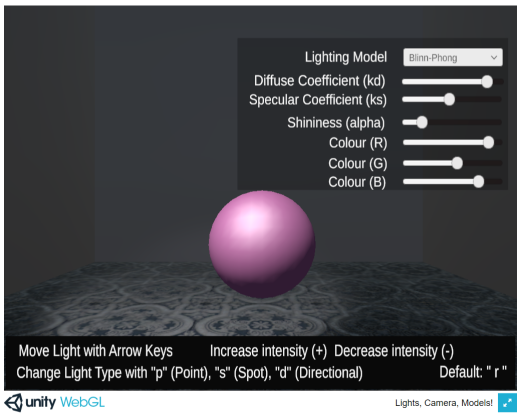

Figure 6: Default Scene with Lighting Model changed to Blinn-Phong

Input: $k_s = 1$

Output: Scene render with Blinn-Phong lit and brighter and larger specular reflection.

Test Case Derivation: $I_s = k_s \cdot i(p, p_0) \cdot \max(0, (L_r \bullet V))^{\alpha}$. Final colouring of any point in a scene is $(I_a + I_d + I_s) \cdot LIGHT\_COLOUR$, therefore changes to the k_s impact the specular component of the final scene.

How test will be performed: Default scene is loaded. Testing framework automatically assigns k_s the new value. Tester manually checks the output images to compare the change in specular reflection.

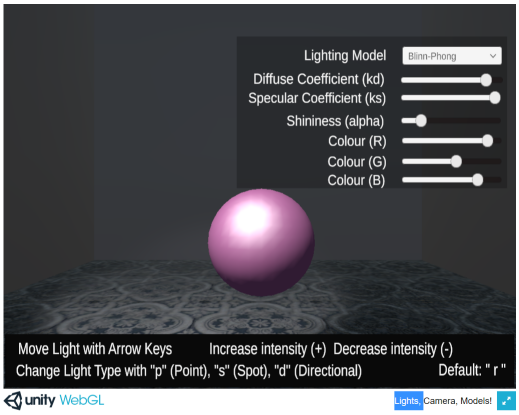

Figure 7: Default Scene with Lighting Model changed to Blinn-Phong and k_s set to 1

2. objMaterialPropChange-valid-kd

   Control: Manual

   Input: $k_d = 0.5$

   Output: Scene render with Blinn-Phong lit, with colour of sphere being dull.

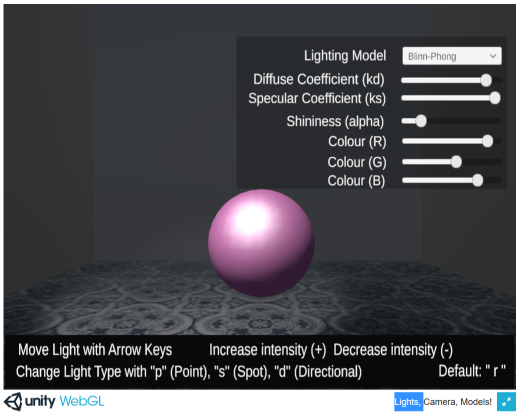

Figure 8: Default Scene with Lighting Model changed to Blinn-Phong and k_s set to 1

   Test Case Derivation: $I_d = k_d \cdot i(p, p_0) \cdot \max(0, (L_i \bullet N))$. Final colouring

of any point in a scene is $(I_a + I_d + I_s) \cdot LIGHT\_COLOUR$, therefore changes to the k_d impact the diffuse component of the final scene.

How test will be performed: Valid scene is loaded. Testing framework automatically assigns k_d the new value.

3. objMaterialPropChange-valid-α

   Control: Manual

   Input: $\alpha = 100$

   Output: Scene render of Blinn-Phong lit sphere and smaller sharper specular reflection.

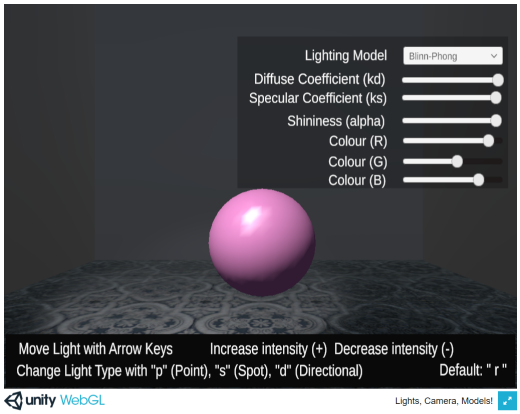

Figure 9: Default Scene with Lighting Model changed to Blinn-Phong and α set to 100

   Test Case Derivation:

   How test will be performed: Valid scene is loaded. Testing framework attempts to assign α the new value.

4. objColour-valid-base

   Control: Automatic

   Input: New (r,g,b) value for BASE_COLOUR picked from GUI picker = (0,0,0).

Output: Scene render of Blinn-Phong with a black object material. All diffuse terms need to be recalculated.

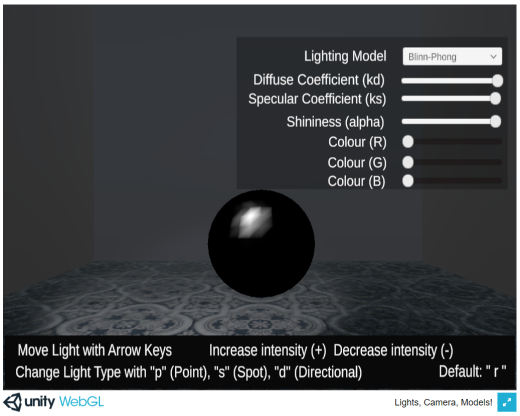

Figure 10: Default Scene with Lighting Model changed to Blinn-Phong and BASE_COLOUR set to (0,0,0)

Test Case Derivation: The BASE_COLOUR at a point is part of the all lighting model calculations. Change the BASE_COLOUR requires recalculating all of the intensity values with this new (r,g,b) information.

How test will be performed: Valid scene is loaded. Testing framework automatically assigns new r,g,b to (0,0,0). System recalculates intensities and recolours object.

Light Changes

1. lightPos-valid

Control: Manual

Input:
Variation 1: Light Type = Spotlight, Press left arrow 5 times
Variation 2: Light Type = Point Light, Press left arrow key 5 times
Variation 3: Light Type = Directional, Press right arrow key 5 times
Variation 4: Light Type = Spotlight, Press right arrow key 10 times
Variation 5: Light Type = Point Light, Press right arrow key 10 times

Output: Scene render based on new position of light source.

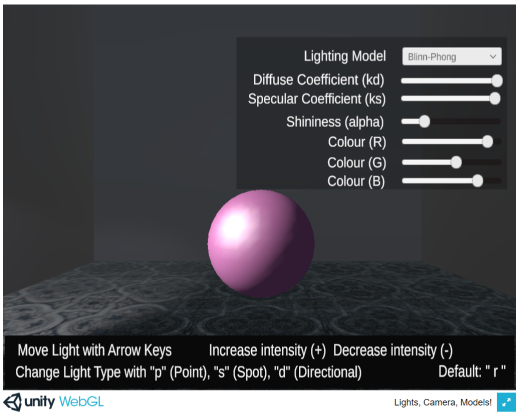

Figure 11: Output Variation 1

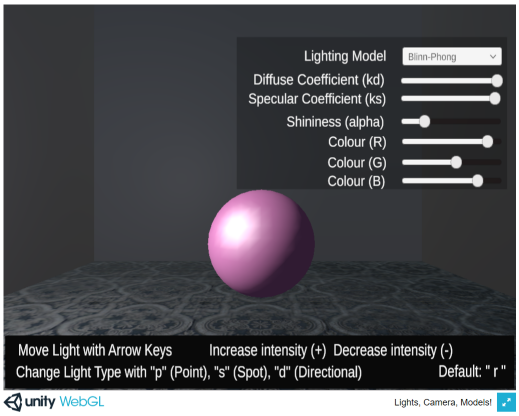

Figure 12: Output Variation 2

Test Case Derivation: Lighting is dependent of position of object relative to the light source; therefore movement in light source position changes the lighting of objects causing all the intensities to be recalculated and the object to be recoloured.

How test will be performed: Valid scene is loaded. Testing framework automatically assigns new position to (2,0,0). System recalculates intensities and recolours moved object.

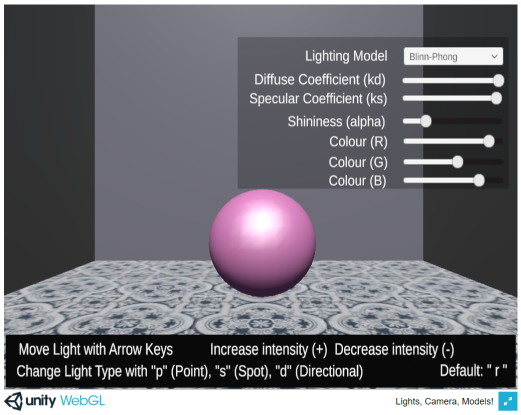

Figure 13: Output Variation 3

[The boundary conditions check and enforce that the light cannot move outside of the scene room in the code. As such, checking the boundary cases here should ensure that no invalid values are allowed. —SS]

2. lightPos-boundaries

   Control: Manual

   Input:
   Variation 1: Light Type = Spotlight, Press right arrow key 31 times
   Variation 2: Light Type = Point Light, Press right arrow key 31 times
   Variation 3: Light Type = Directional, Press right arrow key 31 times
   Variation 4: Light Type = Spotlight, Press left arrow key 20 times
   Variation 5: Light Type = Point Light, Press left arrow key 20 times

   Output: Scene render and lit with Blinn-Phong lighting model, and reflections changing depending on the position of the light.

   Test Case Derivation: Constraints on light position said $0 \ll x \leq SCENE\_HEIGHT$, which means that the centre is on the wall of the scene. This means part of the light would be rendered outside the scene.

   How test will be performed: Valid scene is loaded. Testing framework automatically assigns new position to (0,10,0). System throws an error.

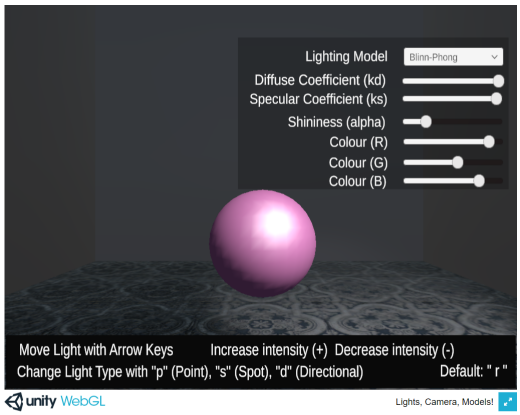

Figure 14: Output Variation 4

lightShape-valid

Control: Manual

Input:
Variation 1: Press s
Variation 2: Press p
Variation 3: Press d

Output: Scene render Blinn-Phong lighting model. Intensities and incidence rays are recalculated depending on the type of light source.

Test Case Derivation: Different types of light project light rays differently. As such the system needs to adapt to the changing type of lights available

How test will be performed: Valid scene is loaded. Testing framework automatically assigns new light type. System recalculates intensities and recolours new object.

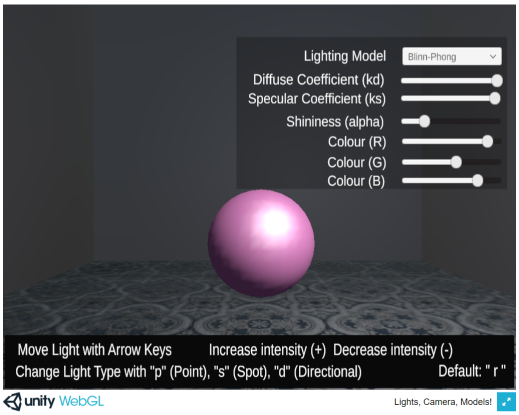

Figure 15: Output Variation 5

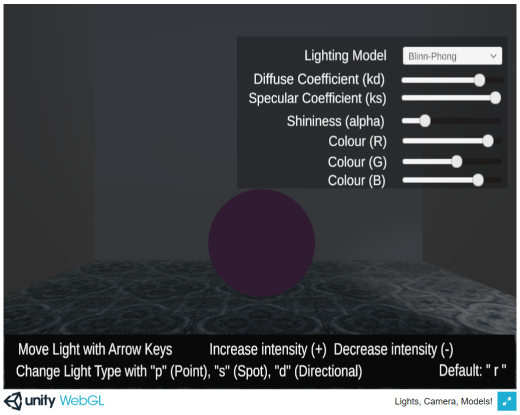

Figure 16: Output Variation 1

4 Nonfunctional Requirements Evaluation

4.1 Usability

5 Comparison to Existing Implementation

This section will not be appropriate for every project.

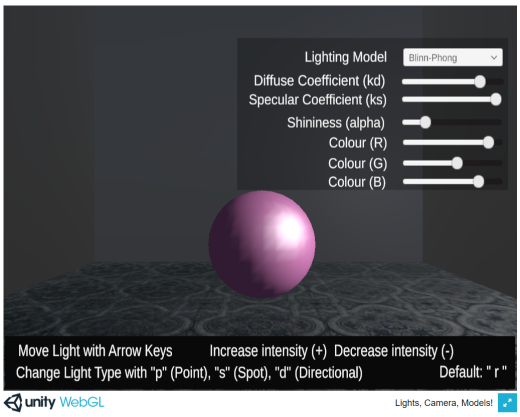

Figure 17: Output Variation 2

6 Unit Testing

7 Changes Due to Testing

8 Automated Testing

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics

References

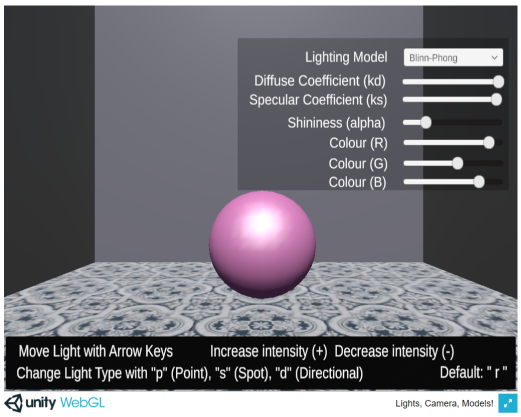

Figure 18: Output Variation 1

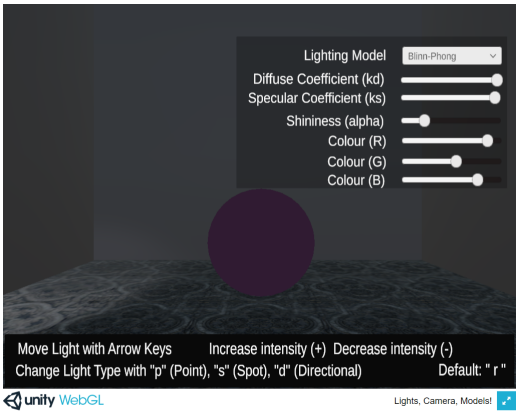

Figure 19: Output Variation 4

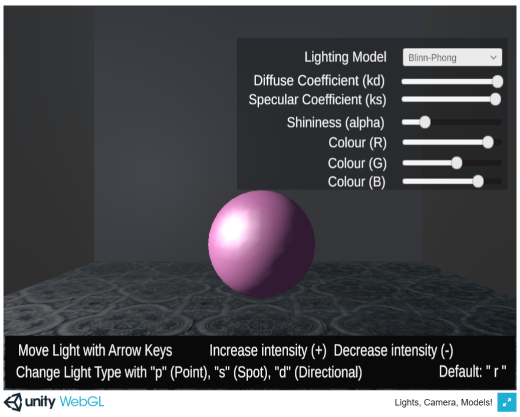

Figure 20: Output Variation 5

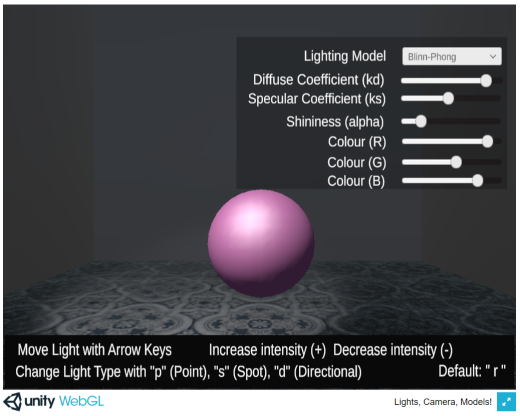

Figure 21: Output Variation 1

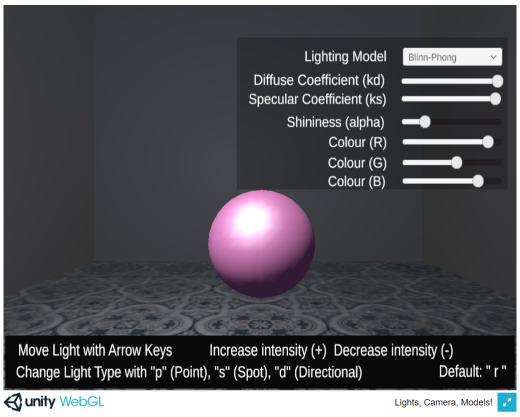

Figure 22: Output Variation 2

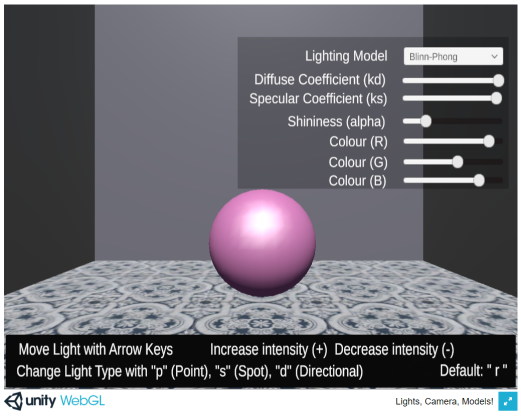

Figure 23: Output Variation 1