Our goal is to create lighting simulation software that is practical and useful for theatrical lighting designers, and can be used interactively.

- Allow the user to easily load and position 3D models within a scene.
- Provide an interactive and easy to use interface to focus and color lights as well as set light intensities.
- Provide an interactive and reasonably accurate renderer to assist in writing light cues.
- Provide access to a highly accurate renderer for producing final renderings.

**Data Model**

- The contents of a scene are stored using a hierarchy of Python classes.
  - The Entity class manages functionality common to both 3D objects and lights. It contains:
    - The position and size of the entity within the scene – translation, rotation and scaling transformations.
    - Bounding box support to highlight the selected object.
    - Picking support to allow object selection using the mouse.
  - The Mesh class manages a single 3D object. It contains:
    - A list of polygons storing the object’s geometry.
    - A set of materials for coloring the object’s polygons.
    - Support for subdividing the mesh for better OpenGL lighting.
    - Support for drawing the object in the scene.
  - The Light class manages a single light source. It contains:
    - An instrument type which models lighting instrument properties such as beam angle, lamp brightness, light distribution and color temperature.
    - A gel color which combined with the color temperature determines the color of the light.
    - A channel number for controlling the intensity of the light.
    - Support for drawing a representation of the lighting instrument in the scene.
    - A scene object contains all of that scene’s entities, and has methods to support adding and removing entities as well as drawing and picking.

**Real-time Rendering**

- We designed a parameterized radiosity renderer supporting real-time adjustments to light intensities, light colors, and the camera viewpoint. We achieve interactivity by dividing the rendering into two phases.
  - The radiosity computation phase determines how each light source affects every location in the scene. It must be run after any changes to scene geometry or light focus. It runs slowly (often on the order of several minutes) and is implemented in C++.
  - Several passes can be run to simulate multiple bounces of light.
  - We accelerate radiosity computations using the GPU, by drawing the scene from the point of view of each patch and performing a hemicube approximation.

**Conclusion**

- Our parameterized radiosity renderer sets this program apart from other existing software and demonstrates the practicality of using this type of rendering in a lighting visualization context.
- The user interface allows the user to quickly configure scenes and lights, and instantly observe the results of their changes.
- We hope this software will be useful as a tool in the Theatre Arts department’s Concepts of Lighting Design class, and as an interactive demonstration of radiosity rendering for computer graphics students.

---

**Interactive Lighting Simulation for Theatrical Lighting Design**

**Abstract**

- The graphical user interface is split into two tabs, each specialized for one task.
  - The focusing tab allows the user to add and remove lights and objects, as well as control their properties and position within the scene. It contains:
    - An orthographic view from above in which the user can select and move entities around the scene.
    - A spotlight view from the viewpoint of the currently selected spotlight, which is used to focus the light using the mouse.
    - A camera view from any point in the scene, which can be changed using the mouse.
    - A property settings panel for the selected entity in the scene, with options such as rotation and scaling for mesh objects, and instrument type and gel color for lights.
    - A toolbar with buttons for adding meshes and lights to the scene.
  - The radiosity tab allows the user to change light intensities, as well as view rendered output images. It contains:
    - The OpenGL view is inaccurate, but is updated nearly instantaneously whenever an entity is moved or an entity’s properties change.
    - The radiosity view is more accurate and is updated nearly instantaneously when a light’s intensity or color changes, but needs to run slow radiosity computations after geometry changes.
    - The radiosity settings panel provides control over radiosity settings such as number of passes, hemicube resolution, and subdivision amount.
    - The Radiance view uses the most accurate renderer, but needs to be redrawn from scratch after any change to geometry or lighting.
    - The light intensity sliders provide a convenient and familiar interface for setting light intensities.

**User Interface**

- The contents of a scene are stored using a hierarchy of Python classes.
  - The Entity class manages functionality common to both 3D objects and lights. It contains:
    - The position and size of the entity within the scene – translation, rotation and scaling transformations.
    - Bounding box support to highlight the selected object.
    - Picking support to allow object selection using the mouse.
  - The Mesh class manages a single 3D object. It contains:
    - A list of polygons storing the object’s geometry.
    - A set of materials for coloring the object’s polygons.
    - Support for subdividing the mesh for better OpenGL lighting.
    - Support for drawing the object in the scene.
  - The Light class manages a single light source. It contains:
    - An instrument type which models lighting instrument properties such as beam angle, lamp brightness, light distribution and color temperature.
    - A gel color which combined with the color temperature determines the color of the light.
    - A channel number for controlling the intensity of the light.
    - Support for drawing a representation of the lighting instrument in the scene.
    - A scene object contains all of that scene’s entities, and has methods to support adding and removing entities as well as drawing and picking.

**Real-time Rendering**

- We designed a parameterized radiosity renderer supporting real-time adjustments to light intensities, light colors, and the camera viewpoint. We achieve interactivity by dividing the rendering into two phases.
  - The radiosity computation phase determines how each light source affects every location in the scene. It must be run after any changes to scene geometry or light focus. It runs slowly (often on the order of several minutes) and is implemented in C++.
  - Several passes can be run to simulate multiple bounces of light.
  - We accelerate radiosity computations using the GPU, by drawing the scene from the point of view of each patch and performing a hemicube approximation.

**Conclusion**

- Our parameterized radiosity renderer sets this program apart from other existing software and demonstrates the practicality of using this type of rendering in a lighting visualization context.
- The user interface allows the user to quickly configure scenes and lights, and instantly observe the results of their changes.
- We hope this software will be useful as a tool in the Theatre Arts department’s Concepts of Lighting Design class, and as an interactive demonstration of radiosity rendering for computer graphics students.

---

**Interactive Lighting Simulation for Theatrical Lighting Design**

**Abstract**

- The graphical user interface is split into two tabs, each specialized for one task.
  - The focusing tab allows the user to add and remove lights and objects, as well as control their properties and position within the scene. It contains:
    - An orthographic view from above in which the user can select and move entities around the scene.
    - A spotlight view from the viewpoint of the currently selected spotlight, which is used to focus the light using the mouse.
    - A camera view from any point in the scene, which can be changed using the mouse.
    - A property settings panel for the selected entity in the scene, with options such as rotation and scaling for mesh objects, and instrument type and gel color for lights.
    - A toolbar with buttons for adding meshes and lights to the scene.
  - The radiosity tab allows the user to change light intensities, as well as view rendered output images. It contains:
    - The OpenGL view is inaccurate, but is updated nearly instantaneously whenever an entity is moved or an entity’s properties change.
    - The radiosity view is more accurate and is updated nearly instantaneously when a light’s intensity or color changes, but needs to run slow radiosity computations after geometry changes.
    - The radiosity settings panel provides control over radiosity settings such as number of passes, hemicube resolution, and subdivision amount.
    - The Radiance view uses the most accurate renderer, but needs to be redrawn from scratch after any change to geometry or lighting.
    - The light intensity sliders provide a convenient and familiar interface for setting light intensities.

**User Interface**

- The contents of a scene are stored using a hierarchy of Python classes.
  - The Entity class manages functionality common to both 3D objects and lights. It contains:
    - The position and size of the entity within the scene – translation, rotation and scaling transformations.
    - Bounding box support to highlight the selected object.
    - Picking support to allow object selection using the mouse.
  - The Mesh class manages a single 3D object. It contains:
    - A list of polygons storing the object’s geometry.
    - A set of materials for coloring the object’s polygons.
    - Support for subdividing the mesh for better OpenGL lighting.
    - Support for drawing the object in the scene.
  - The Light class manages a single light source. It contains:
    - An instrument type which models lighting instrument properties such as beam angle, lamp brightness, light distribution and color temperature.
    - A gel color which combined with the color temperature determines the color of the light.
    - A channel number for controlling the intensity of the light.
    - Support for drawing a representation of the lighting instrument in the scene.
    - A scene object contains all of that scene’s entities, and has methods to support adding and removing entities as well as drawing and picking.

**Real-time Rendering**

- We designed a parameterized radiosity renderer supporting real-time adjustments to light intensities, light colors, and the camera viewpoint. We achieve interactivity by dividing the rendering into two phases.
  - The radiosity computation phase determines how each light source affects every location in the scene. It must be run after any changes to scene geometry or light focus. It runs slowly (often on the order of several minutes) and is implemented in C++.
  - Several passes can be run to simulate multiple bounces of light.
  - We accelerate radiosity computations using the GPU, by drawing the scene from the point of view of each patch and performing a hemicube approximation.

**Conclusion**

- Our parameterized radiosity renderer sets this program apart from other existing software and demonstrates the practicality of using this type of rendering in a lighting visualization context.
- The user interface allows the user to quickly configure scenes and lights, and instantly observe the results of their changes.
- We hope this software will be useful as a tool in the Theatre Arts department’s Concepts of Lighting Design class, and as an interactive demonstration of radiosity rendering for computer graphics students.