Objective:
This game is created in light of the recent power crises our nation is facing. It is intended for students at the UWB campus hoping to remind all to conserve energy at school and at home.

In the game, the player controls the Lamp (its hero object) to collect energy particles and power items in the environment. The purpose of the Lamp is to go from room to room (after hours) and collect as much energy as possible; plus it is to turn off all unused electronics in the building.

The Lamp’s work does not go un-noticed. There are many Power Bugs that feed off of these power lines. They can also take energy away from the Lamp if they touch the Lamp. Therefore, the Lamp must destroy these bugs as it sees them and the Lamp must avoid touching them.

The Lamp will succeed if it turns off all unused machines in the building. However, the Lamp will fail if it ran out of power before the task is accomplished.

How To Play:
The game is very interesting to play. You control the Lamp with the mouse or the keyboard. Your goal is to kill as much many bugs as you can and not get killed. The more bugs you killed, the more points you will obtain. The following will show you in details of hot to control the Lamp.

Lamp Control:
Lamp can be controlled through the Keyboard or the mouse: I strongly suggest using the mouse. The control keys are as follows:

Key Board Settings:
D - rotate head clockwise
F - rotate head counter clockwise
Space bar - fire bullets
Use arrow keys to move Lamp
P - Pause Game
R - Reset Game
Q - Quit Game

The mouse can also be used to control the Lamp. The table below will lay out that you can and can’t do with the mouse:
**Game View**

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMBD</td>
<td>Turn Lamp’s head to this direction.</td>
</tr>
<tr>
<td>LMBDr</td>
<td>Lamp’s head follow mouse pointer</td>
</tr>
<tr>
<td>LMBU</td>
<td>Fire 1 bullet.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMBD</td>
<td>Define target point for Lamp to move to.</td>
</tr>
<tr>
<td>RMBDr</td>
<td>Same as RMBD</td>
</tr>
<tr>
<td>RMBU</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game View</td>
<td>Define focus point for zooming</td>
</tr>
<tr>
<td>World View</td>
<td>Zoom</td>
</tr>
<tr>
<td></td>
<td>Zooming ends</td>
</tr>
</tbody>
</table>

**World View**

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMBD</td>
<td>Define target point for Lamp to move to.</td>
</tr>
<tr>
<td>RMBDr</td>
<td>Same as RMBD</td>
</tr>
<tr>
<td>RMBU</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game View</td>
<td>Flag Panning action</td>
</tr>
<tr>
<td>World View</td>
<td>Pan Game View</td>
</tr>
<tr>
<td></td>
<td>Pan action ends</td>
</tr>
</tbody>
</table>

Notes: LMBD = Left Mouse Button Down  
LMBDr = Left Mouse Button Drag  
LMBU = Left Mouse Button Up  
RMBD = Right Mouse Button Down  
RMBDr = Right Mouse Button Drag  
RMBU = Right Mouse Button Up

*Do you get all that?*

**Fire Limits:**

Lamp can fire no more then 10 bullets within the viewable area of the Game View. A bullet will die/expire when it is outside of the Game View’s bound. When this max number is reached, Lamp will not be able to shoot anymore bullets until one or more of the lived bullets died.

Note: Holding the Spacebar will tell Lamp to fire continuously. This may make Lamp, at times, seemed to be helpless. Try balancing the number of bullets with the screen size you’re playing. Save one or two incase a bug comes too close to you.

**Vision Limits:**

Because all circBugs are created with random colors, some may end up having the color of the game’s background. These bugs are the ones you can’t see. That’s why one of the items in the “Cool Tools” for Lamp is a “Supper Vision”. This allows Lamp to see object’s critical points (i.e.: their bounds). Try turning this one when the game is empty you have not yet advanced to the next level.
User Interface:

Here is a closer look at the menus on the right hand side:

This Screen Show the status of the game. It has Lamp’s Energy level, fire type, Speed and level information.
This screen shows what you can do in the game. Options you can choose are:

1) Pause
2) Start/Restart a game
3) Get Help – which shows you the control settings using the Keyboard
4) View the Top 10 scores. Only available at introduction scene or game over scene.
5) Exit and quit the game.
6) Can select and manually play around with the lamp’s parts…try it…

Game Screen:
Below is a series of screenshots of the game in play. The screens are shown in this order:

1) Introduction scene
2) Actual game playing scene
3) Game Over scene

Introduction
This is the introduction screen to the game.
Playing Scene:

This view is set to maximum allowable size for a Game View – which is 200x200. The red and yellow objects are your bullets and the circle objects are the Circbugs.

Noticed the light green bug on the bottom right corner, it is touching the wall on this frame, it will bounce off and move at a different direction on the next frame.

Hint: With the combine use of the spacebar to fire and the right mouse button, you can chase after a bug.

While in game Play, you can request for help. This is what you’ll see:

**Key Board Settings:**
- D - rotate head clockwise
- F - rotate head counter clockwise
- Space bar - fire bullets
- Use arrow keys to move Lamp
- P - Pause Game
- R - Reset Game
- Q - Quit Game
**Next Level Scene:**
This is a scene you’ll see when you advanced to the next level:

![Next Level Scene](image1)

**GameOver Scene:**
This is the scene you’ll see when your energy runs out... I’m sure you don’t want to see this too often.

![GameOver Scene](image2)
System Architecture:
This program is written using VC++ 6.0. The main hierarchy of the program looks like this.

As a supper class, TgraphicsObject bears the most work. Similar for Levels and TxformInfo class. Most of the work for these classes are up in the supper classes which makes the base class really simple.
**Limitations of this version:**

This version of the game has been challenging. The most challenging part was to get the data structure to support game’s need. The current data structure needs some remodifications.

1) It needs a better way of separating out all the math involves in collisions. I was not able to do tight bounding or accurate collision checking. My hero object and supporting object sometimes overlap momentarily or worst, my supporting object goes right through my hero object. Right as it stands, when my hero object touches something moving, it move backward to where it was one frame plus three more pixels. My supporting object does the same thing.

2) It needs have better handling with state transition (e.g.: going between in intro scene to the game scene and the game over scene…)

3) It need better support for type verification. That is, each object knows what its type is. As of right now, the structure above provide some of this.

**Next Release Feature Lists**

For the next release of this game I would like to do the following (either from scratch or improve in it):

1) Tight bounding of objects
2) Better collision detection and handling.
3) Add texture and sound as appropriate as possible
4) Implement more levels
5) Provide collectible items for the Hero object such as weapons and energies
6) Make game transition between states of game play smoother instead of as abruptly as it is now.