DUNGEON CRAWL UPDATES

By Ronin Rodkey and Will Christians

The program did not change too much mechanically, as the prototype's mechanics were sufficient for implementing everything else we needed.

The greatest change to the program is the complete graphical overhaul. Image files were accessed and incorporated with java's imageio library. The old graphics were dull and uninspired, but now they are bursting with flavor like a well-marinated block of tofu. Additionally, the program gained some new fancy features:

Menu System:

We wanted the game to not just throw you into the dungeon when you press enter on the console. Adding then menu allows the user to mentally prepare before delving into the weird and wacky world of DUNGEON CRAWL. In the menu, you can access leaderboards, view a rules screen, and choose the game mode (endless vs. timed). The menu is navigated through keyboard, because we already knew how do that.

Leaderboard:

The leaderboard holds the list of top scores for the timed mode. It is accessed via FileReader and changed via PrintWriter. This allows the user to keep track of their scores automatically, without need for pen and paper.

Time System + Associated UI:

The time system allows pressure to be placed on the player. They now have a motivation to move. We accomplished this by adding a variable to the MazeGame class which is increased by a 60th of a second every frame.

The Chaser:

The chaser is a scary boy who moves faster and faster as the player get deeper. Imbued with powers of darkness, he chases the player down and steals their flashlight, losing the player points process. He brings the 'horror element' to the game. Essentially he is just another sprite that follows the character's movements.

Points System + Associated UI:

The points system allows the player to tell how good/bad they are playing the game. The score increases like the time variable, and also increases when the level increases.

Gradient Blindness and Rendering:
When implementing graphics, the game ran at a snail's pace when the map got large. By only drawing a certain amount of tiles, the game now runs quickly, regardless of the level. An initial method assigns a value to each subtile regarding its visibility based on the location of the player. Then, only the subtiles the player sees are drawn in the map. It's like the old saying goes: if a tree falls in the forest, and nobody's around to see it, then you don't need to draw it. Finally, when the darkness is drawn around the player, a translucent rectangle is drawn around the tiles farther from the player--making them appear dim, but not invisible.

"Light" Subtiles:
When you stand on certain tiles, the whole map lights up! This was fairly simple to implement: when the player stands on one of the green subtiles, it sets each subtile's blindness level to 0, rendering the whole map visible to the player.