I. Game Idea Summary

A. The basic idea for our game is to create a hybrid between puzzle games and platformers in a similar manner to games such as Portal and Shift. Our game is based around light and the interaction of colors which create various effects in the gamespace. The base mechanics revolve around three overhead lights each of which will shine a different color (e.g. R, G, B) down onto the playable space (think spotlights). It is possible that the player will be able to rotate and move these lights to illuminate different areas in the playable field. The player will use these lights not only to see where they can move their avatar, but also to interact with the environment (e.g. a "red" platform will only be solid while a red spotlight is shining on it). Further, the player will have to mix and match colors (e.g. mixing the Red and Blue lights to create a purple light to interact with purple objects and platforms). The player will have to manipulate the lights as well as other objects such as mirrors and prisms (puzzle aspect) to create a way of moving their avatar from a starting position to a goal position (platformer aspect) within each level. An outline of the ideas we came up with while brainstorming follows:

II. Whiteboard Drawings

A. Ideas1

B. Ideas2

III. Game Ideas

From Joe Kohlmann, Dan Szafir and Sam Wasmundt.

A. Genre

1. Hybrid between puzzle and platform.
2. You get to take part in the level design. You get to decide how this becomes a solvable level.
3. Educational aspect: teaches about colors, light physics, etc.

B. Mechanics

1. Three overhead lights
   a. Default: Red, Green, Blue
   b. Later: Other lights that combine differently?

c. Does the player get to control the position of these?

Maybe we introduce gradual control over the

d. These lights combine to reveal objects that are different colors.

2. Tools to manipulate the lights:

a. Prism

b. Mirror

c. Light Sensors (shine light on these to activate other things)

d. Light Circuits (activate objects / make others visible)

e. Disco Ball: Scatters the lights or lights up the whole level (for a time)

3. Environmental effects change light dynamics

a. Fog: scatters the light

b. Water: a pool reflects light (but scatters it based on how turbulent it is)

4. Player only sees certain wavelengths.

Levels and strategies change if player can see in the infrared or ultraviolet spectrum.

a. Visibility changes based on:

i. Health

ii. Power-Ups

iii. Levels Themselves

5. Player has three color-coded health bars.

a. These would be Red, Green, Blue. The appropriate bar decreases if the player is in direct contact with a beam of that color.

b. Player absorbs the light he/she is standing in, so that light depletes if they idle / spend a lot of time in the light.

c. Maybe the player *is* a certain color.

d. Maybe the environment reacts differently (reward or penalty-wise) based on whether player color and object color is complementary, supplementary, the same, etc.

Maybe that's easier than the player only being able to see certain colors. Instead you can only work with certain colors.

C. Penalties

1. Objects that can hurt.
2. Fall into an abyss if you miss a platform or something.
3. What if you could only stay in direct contact with light for a certain amount of time?

D. Goal/s

1. Progress to the next level.
   a. Maybe you get shot off to the next level as a photon.
2. Acquire Power-Ups through exploration.

E. Burning Questions

1. Do we need enemies? Is player vs. environment sufficient?
2. What kind of scale do we want for levels?
3. How does a color-blind player play this game?

   We'd rather get a good game out first. If nothing jumps out to us as far as how to solve this issue, that's fine.

IV. Game Technologies

A. JavaScript, jQuery, WebGL, GLSL

Well, you gotta use these!

B. Three.js

Someone made a 3D engine in JavaScript? Awesome.

C. Custom Level Loader

Something we'd build to load some kind of level file.

D. To Investigate

1. Web Workers

   This standard is essentially an approach to multi-threading in JavaScript.
2. Some Sort of Model Loader

   Looks like Three.js has some kind of model loader, which is good.
3. Sylvester Vector Library

   Defines some vector and matrix classes for JavaScript. Three.js might obviate the need for this.