# Game Analysis

*By Chinmay Patwardhan (cmp49)*

## Technical Review:

Time Review: I began working on this project on this project on the night of August 29th and finished on the night of September 3rd. In total I estimate that I spent 12-15 hours on it. I dedicated my first day to playing around with a JGame sample project and seeing what JGame was capable of. I then decided on a design for a game. Originally I was hoping to do something golf related, but I realized JGame would be a poor framework for what I had in mind. After deciding on a game design, I spent roughly four hours coding to build the basic layout for my first stage and got almost nowhere. Getting my physics to work the way I intended took a lot of time. After that though, everything was smooth sailing and I didn't hit many bugs until I had to learn how to define a bounding box. (I had to do this for my Darkness object.) I also spent a significant amount of time refactoring code to use constants and split my separate classes into separate files for readability. The last big time consumer was setting up the transition of game states.

Planning: My original plan consisted of 8 stages, but I did not allot time for them. Step 1 by far took the most time, so in the future I would probably want to start very early.

1. Build a Player and Platform class and implement the game physics. (See the README for a full description of the physics that I chose to implement aside from simple gravity.)
2. Build Fireballs and code the collision with Players and with Platforms.
3. Make playing field wider than the screen and implement side scrolling. Also make a function that allows me to parse the stage from a String.
4. Build Darkness, make it move in a way that never lets the user get more than a set distance in front of it, and code the collision with the user.
5. Build an item class and code the collision with the user.
6. Code the transitioning of "states" to switch to stage 2 (and show title, instructions, and win/loss screen) when the user reaches the end of stage 1.
7. Build stage 2. (Allow user to place Platforms, and then jump to heaven)
8. Implement cheat codes.

I stuck to this plan almost exactly except for one step: 7. Originally, I was going to let the user pick up two types of Items, one would turn into a permanent Platform in stage 2 and one would be temporary. I decided this was too confusing while placing the blocks, so I made all the placed Platforms temporary instead, so that I still coded that functionality. This also made it harder for the user. In the end, I think I build a pretty difficult game… I struggled to win when I played, but it was certainly winnable.

Status: I believe my code accurately and functionally accomplishes my plan.

Readability: The only serious readability flaw that I can find in my code is that a Player doesn't keep track of his own lives, and the main StairwayToHeaven model keeps track of it instead.

This makes no sense to me, but I kept it this way because it was already implemented in the example program. I can also identify two functions whose names aren't necessarily very clear: goToHell() and getRandomItemSteps(). Both of these are one line functions because they helped the way I think, but I realize for others it may have made it less clear. goToHell() just calls lifeLost() and getRandomItemSteps() just returns a random number number of platforms until the next Item is placed. The code could also be more clear and maintainable if I weren't building in a subclass of StdGame because then all the core functionality for the game would appear in the same file instead of in two files. Most importantly, I think JGame very poorly implements collisions because checkCollision(a,b) is different from checkCollision(b,a). This is confusing because some JGObjects don't implement hit() and others do even though all of my objects were involved in at least one type of collision.

Extensibility: The code could easily be extended to allow the user to pick up different types of Items, perhaps power-ups or extra lives. A harder part of the code to extend would be to allow the user to different types of platforms in the build stage, such as springs or moving platforms. Making the game progressively more difficult (more frequent Fireballs, more difficult Platform layouts, etc.) would involve heavily modifying existing code. The game is also currently designed to store lives, but no measurement of health. I think it would take some work to have a user have a set amount of HP and lose HP every time he or she is hit with a Fireball.

Testability: Building the stage is probably the only easily testable feature of my code. Everything else depends on a GUI and human interaction. For building a stage, I could set up test cases to check if the corners of each Platform occupy the expected locations. Other than that, automated test cases would be difficult to make.

Conclusions: The majority of my redesign time was spent refactoring constants. This time could have been cut down if I planned out a Constants file before beginning the rest of the code. I also could have better named a couple functions to make their purpose more clear. I did not find other significant defects in my code, and I am very satisfied with its quality.

Future Work: I think when I completed my game it was still too difficult to win. I tweaked with the constants to make Fireballs and Darkness easier to avoid. If I could work more on the project I would add four things to make the game both easier and more fun:

1. Health: I would give the users HP for each life. This was falling and being caught by Darkness would still kill the Player, but being hit by a Fireball would only make the user lose HP. This would make it easier to get through the first stage and make the game more enjoyable. To implement this, I would simply call a function loseHealth() which would decrement the HP and check if the user had any HP left. If the user runs out, it would call lifeLost().

2. Checkpoints: Every so often the user would pass a checkpoint. Every subsequent death would return the user to the last checkpoint. This would make it easier to get through a very long stage 1. I ended up making stage 1 half as long as initially wanted to because it was too hard for me to maintain my focus long enough to get through the whole thing without dying. Furthermore, this would be the perfect place to make the game more

challenging. Every time a check point is passed, the Fireballs could become more frequent, Darkness could move faster, the Platforms could begin to move… There's a whole world of possibilities that my current design allows for with very few changes.

3. Power-ups: This would be very easy. I could build a class hierarchy extending Item that would call the appropriate function to affect the user. This could be temporary invincibility from Fireballs, more HP, more lives, larger jumps, higher speed, etc. It would be really easy to implement because it would just involve changing fields in the Player class and simple if statements in the hit() function to make the user invincible.

## Code Submission:

See package titled CodeSubmission. I changed the way collisions were handled to make the code more maintainable.

Let's use a hypothetical situation. Say I had two types of Objects: Poles and Cars. When a Car collides with a pole it should cause the Pole to bend() and the Car to dent(). In my old design, I wouldn't have cared whether I implemented this collision in the Car's hit() function or in the Pole's hit() function, but now I realize it's more maintainable if I implement it in **both** classes. In Car's hit() it should call dent(), and in Pole's hit() it should call bend(). This way, I know to look in Car to change the action of a Car during a collision and the same for a Pole. It just makes more sense. In my old design, I might have called pole.bend() from within Car's hit() function.

In addition, I pretend that Players keep track of their own lives.