Mini-Project 4: Interactive Programming

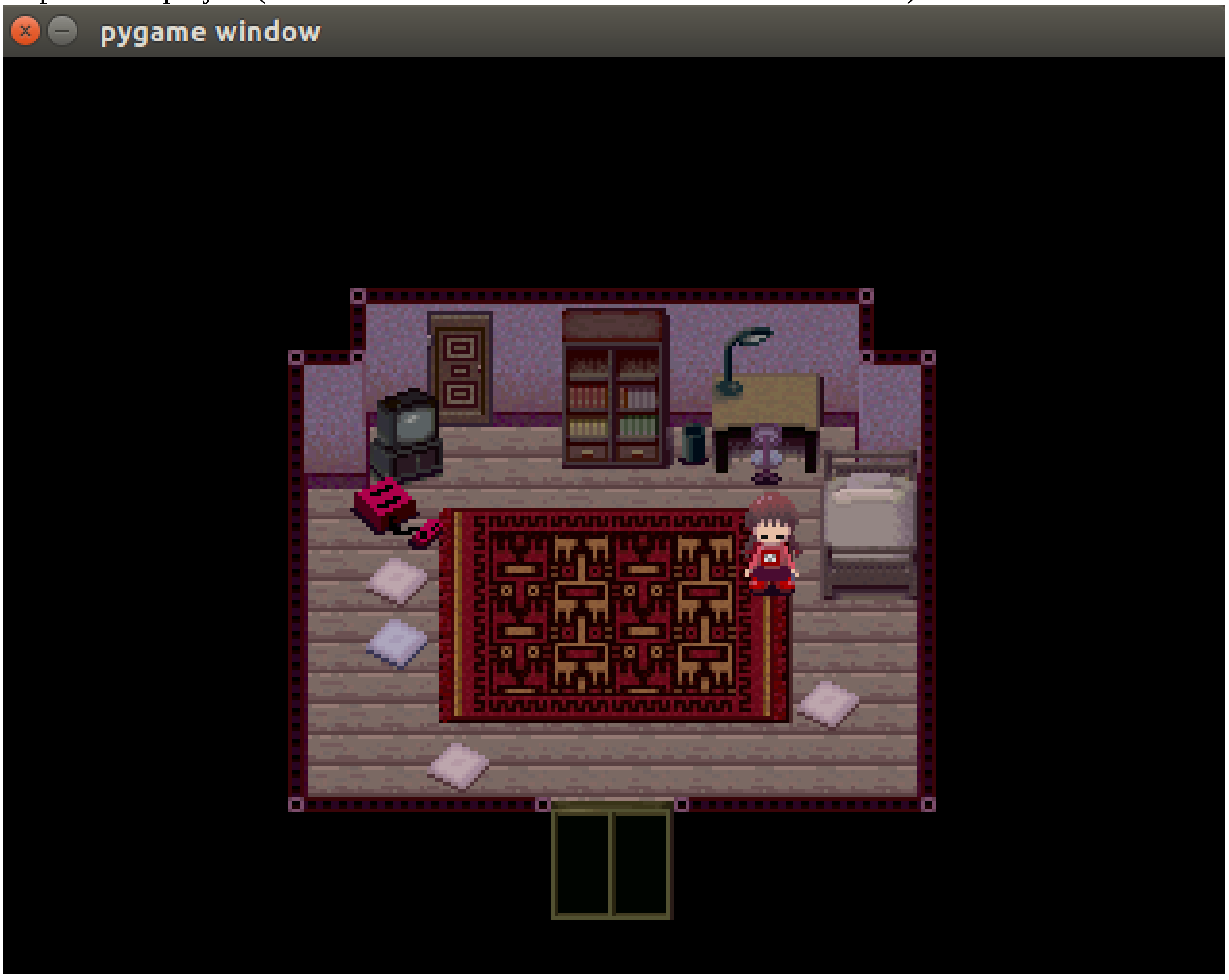
Ian Paul and Taylor Sheneman

Project Overview

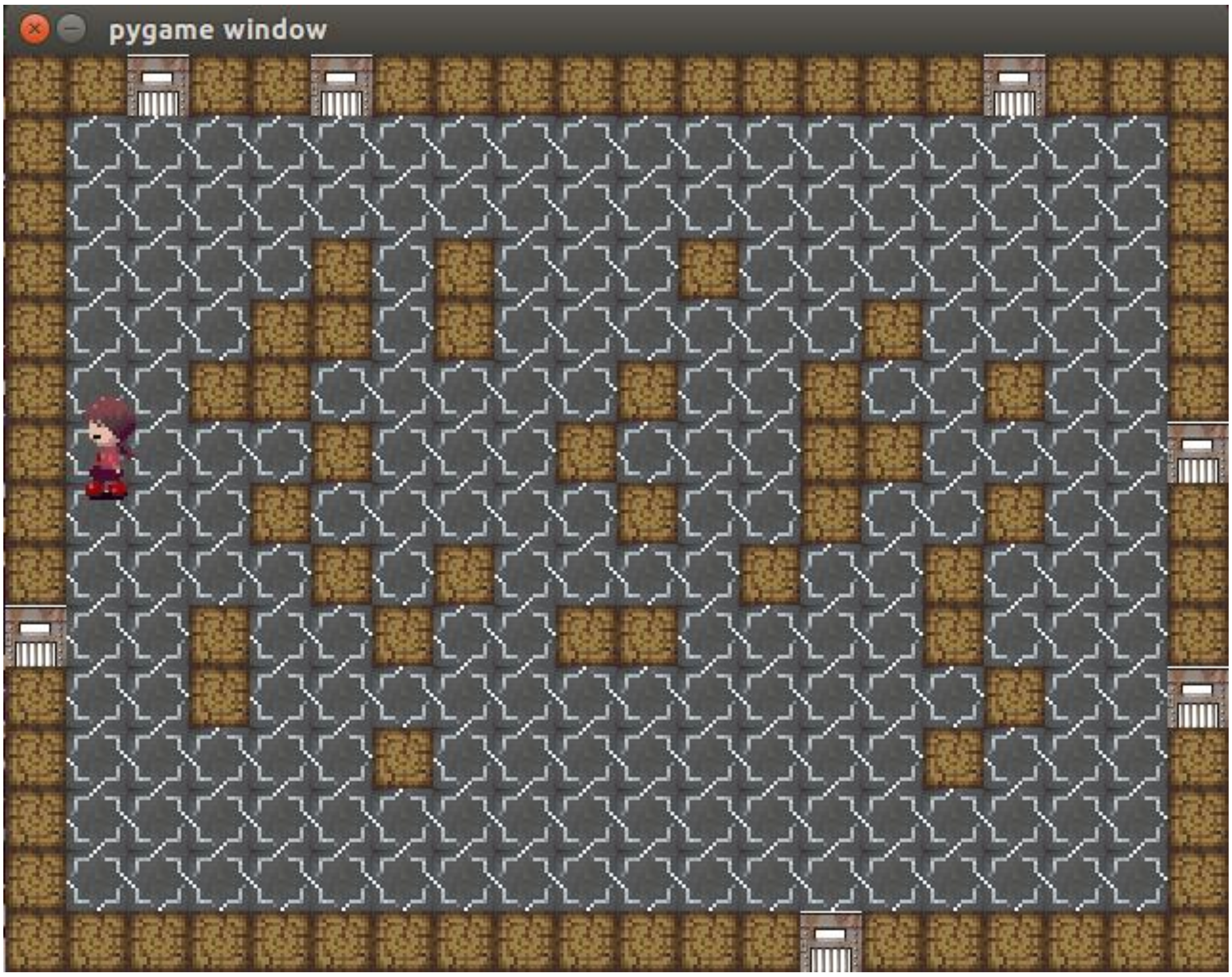
Our goal for this project was to create an atmospheric, randomly-generated exploration game using PyGame. There is no objective, just a surreal, aimless dreamworld to wander around in. You wake up into the dream in your room; upon venturing out, the room is lost, replaced by the endless dreamscape. The only escape is to wake up, back in your room, only to try again...for a time.

Results

We did a lot of interesting stuff in this project. Wall mapping and collisions, spriting and character animations, and random generation are the most interesting. Our final product incorporates art assets loaded as sprites with animations for their motion, collisions between the player and multiple types of objects and randomly generated environments for the player to explore. There is also a bit of a disturbing surprise. It's pretty creepy, and in the theme of the game that inspired this project (and from which we borrowed a number of art assets).



1The Game using other the default room

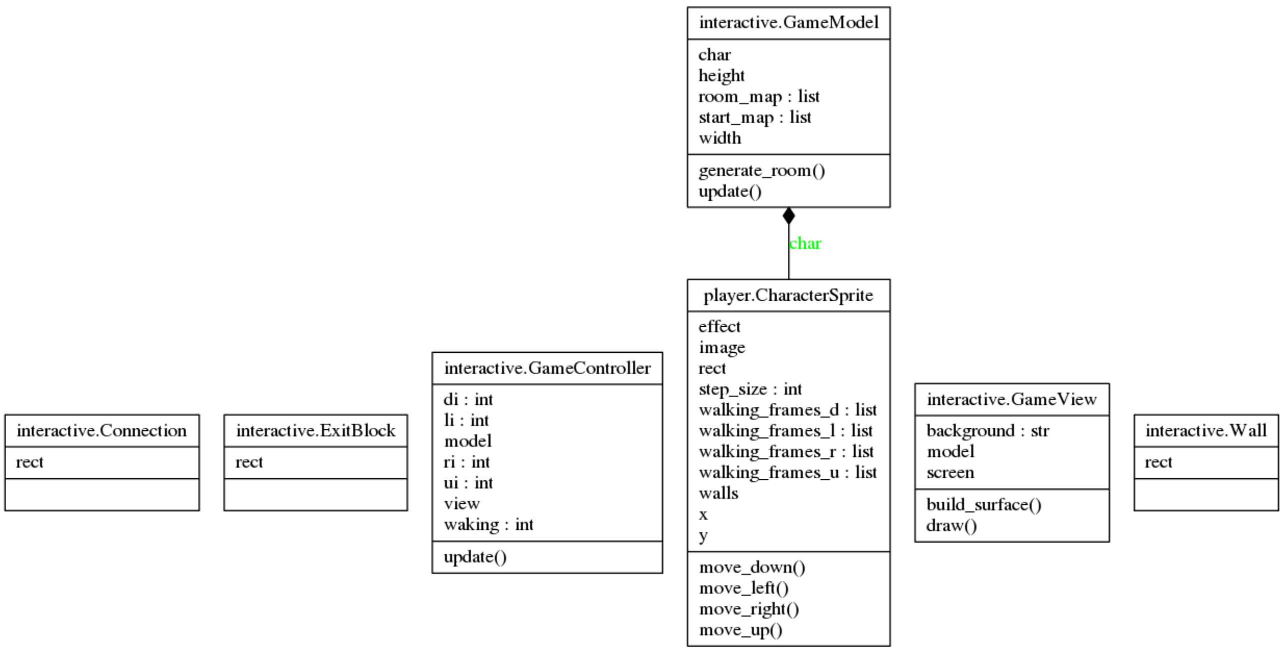


2The Game using a tileset and a generated room

Implementation

The three major components were the Model, View and Controller. Our game ended up passing around much more data between the three of these than we did with the model in class. We made a sprite sheet class, which is how we broke up big images with different sprites in different stages of animation into smaller moving pieces, and classes for the different types of sprite objects mapped within the game. The organization broke down a bit towards the end as we incrementally hacked together bits of our code to interface with itself; there are certainly areas where the code could be tighter (for example, we don't really take advantage of the fact that pygame's Sprite class can include both the image and Rect in neat, compartmentalized fashion).

We use lists to store data about the level. We keep none of it permanently, as a design choice. There was never supposed to be consistency within the dream, and we eventually realized that it would actually make more sense for us to not keep track of the room layout, and just generate as the player wanders. Trying to go back to an earlier room? Too bad, the dream has changed.



3A slightly unhelpful UML

Reflection

We didn't hit any major roadblocks along the way, it was just a very large project and neither of us had too much time. We could improve some of how we modularized our code, there are a couple of places where things would be better as functions and we didn't build them into functions ever. I don't think we appropriately scoped the project: we didn't struggle particularly with the computing in it, there was just more man-hours of work required than we were able to give. Unit testing wasn't a problem, we found it easy to incrementally test our code and work on parts separately. Going forward, not that many of these skills are appropriate. Much of what we learned in unique to PyGame and isn't appropriate for other areas of computing. It has, however, given us a sense for how much of a pain graphical interfaces are to work with.

Teaming was good. It took Taylor a while to get into the swing of things, mostly because of problems with understanding PyGame documentation, but we worked past that alright. We were both busy, but we both still put in a lot of hours by the end of the project. Next time, we would try to scope more appropriately to how much time we had, and try to implement more things earlier.