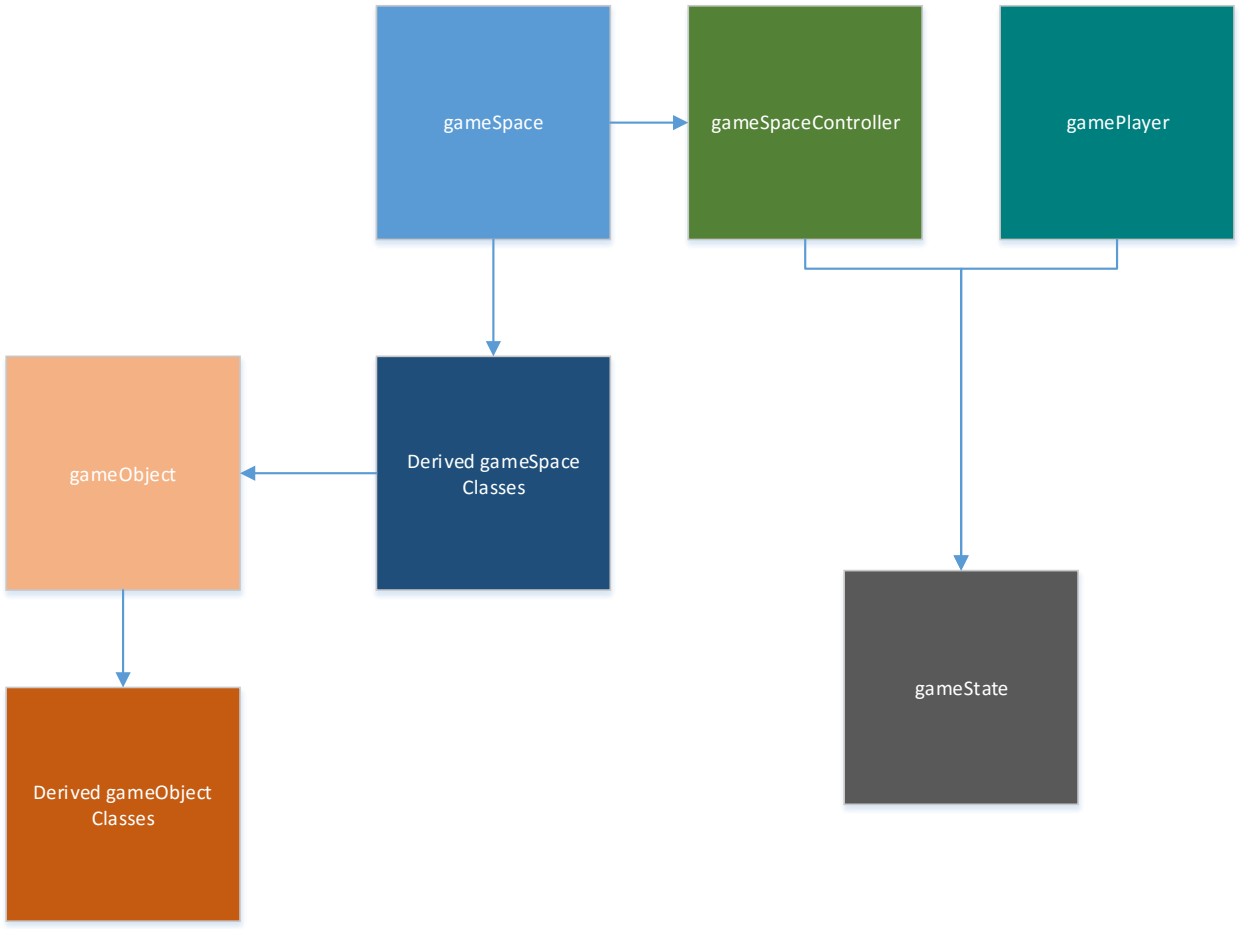
Jeremy Prater

CS 162 – Final Reflections

Class hierarchy



Testing

- I played the game testing all possible game paths.
- I attempted to enter bad user input to locate possible failures and segmentation faults
- Valgrind to verify no memory leaks

Usage

./final [-d]

-d Turn on verbose debugging for objects/player

Reflections

I started with a class hierarchy that initially took much overhead to manage. By breaking functionality out into smaller sub-classes, my logic and code design became much easier.

Using derived classes to extend gameSpace and gameObject were very helpful in creating custom behavior and logic.

I had to implement a class gameSpaceController which allowed interaction between gameObject/gameSpace/gamePlayer. This expanded the user experience by allowing characters in the game to give the players objects with very little code and without hacking around class boundaries by making members public. The common framework between gameSpaces and gameObjects allowed for objects to interact with spaces and for spaces to query the status of objects, or the player to determine behavior.

One example of this is the behavior that Squirtle shows when it only gives you the key after you heal its friend in the river with the Elixer that you have to brew in the teapot at the cabin with the Flowers that you find on the trail.

Given more time I may have unified gameSpace and gameObject into a single parent class and used a more advanced messaging system.

Solution to game is located at the top of gameSpace.cpp

It’s 11:33pm so I’d better submit this now!