Brooklyn DiPietrantonio

Game Programming I

Jeff Meyers

14 December 2020

Final Build: Torches

Game Name: Torches

Game Description Short: Traverse your way through dungeons with nothing but a torch!

Game Description Long: Traverse through a deep, dark dungeon with only a torch in hand! Find various items to help illuminate the way such as oil and lighter fluid, helping you find the key out of the darkness. Watch your step though, as many leaks and puddles may burn out your torch!

Credits:

* Art, Design, and Programming – Brooklyn DiPietrantonio
* Programming help - Jeff Meyers
* Playtesting – Robby Billings, Zach Wojcik

Genre: Puzzler

Controls: WASD or Arrow Keys, 1 to Start

Developer Log:

While creating Torches, I ran into a large amount of familiar issues that I faced in the past while making Unity games. Being able to stretch those muscles again was great for my programming and Unity skills.

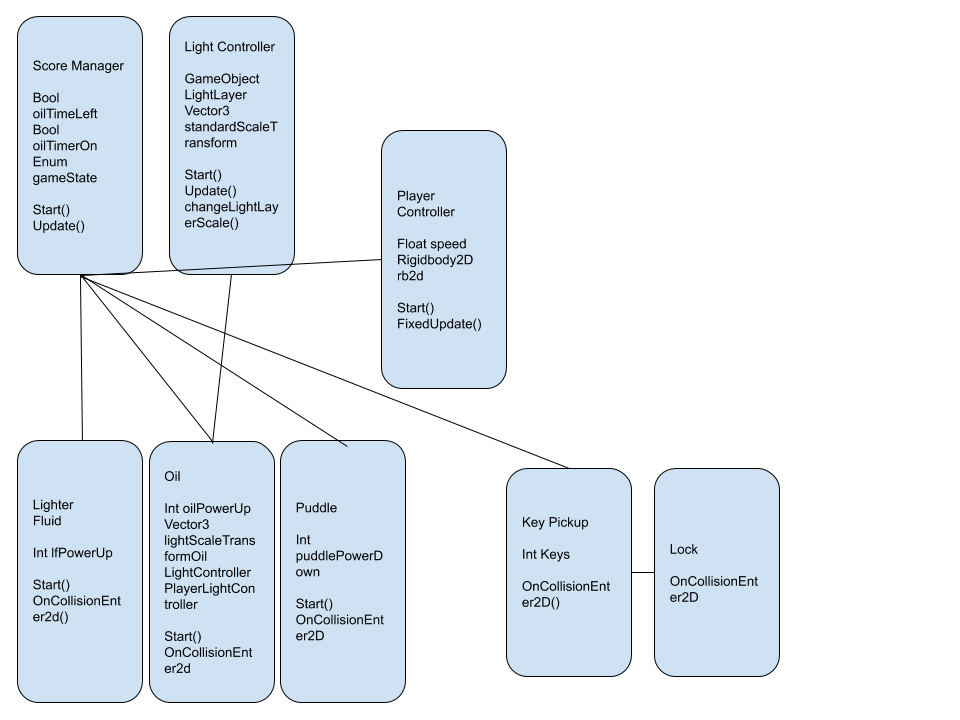
Initially, I used the tagging system for making new items. In the long run, this could have caused me issues as it’s not OOP. Moving forward I made classes for every item as this allowed for better customization, control, and organization of each item. Tags remained for player collision, as this allows for the code to be reused on items without having to name the game object a specific name.

Another issues I experienced was creating the ScoreManager and ensuring it did not have too much responsibility. My first approach to programming always had me consolidating scripting to leave things as simple as possible. Coming into professional programming standards, this isn’t good practice. One of the downfalls I had for my ScoreManager is that it also manages Oil burn time, if I had the time to further push OOP standards, I would make the Oil item manage its own time. This would mean the Oil item would depend on the UI item instead of depending on the ScoreManager and then the UI item.

Something that I felt wrong about doing was the use of an ‘if’ statement in the player controller. This issue stemmed from not wanting the player to move before the timer started. I found this very far into bug testing, so instead of bug testing the <GetComponent> to disable the player controller script on the player in the ScoreManager enum, I added an ‘If’ statement that allows the controls to only run if the gameState enum is set to GameProgressing. It solves my problem with people cheating the timer but causes some issues with separation of concern and code reusability as now the code depends on the ScoreManager.

Something that had been suggested was for pickups to be put into an array in the LightController. In the long run, this would have been very beneficial as it would have allowed for a sort of inventory management. Sadly, time did not allow for this as it would have required rewriting a good chunk of my code. I think if I choose to develop this game further, its something I’ll be adding.

Postmortem:

* Class UML diagram
  + 
* What dependencies does your game have how did you handle them?
  + A lot of scripts were dependent on the ScoreManager script due to the timer style of the game. ScoreManager manages those timers as they interact with each other during gameplay.
* How did your class/components design work out? What would you change if you could start over?
  + I don’t think I would change very much about my design, as for what I want it to do, it works pretty well and feels clean. I think moving forward, if I did change anything, it would be just to allow for further expansion of the game, such as the array in the LightController.
* Rate how reusable the game codebase is? What percentage is specific to the game? What percentage is specific to the platform (monogame/unity)?
  + Most of the codebase is reusable, but mostly only for Unity. Some of the code, such as the menu system, is something that I actually reused from previous projects. This applies so the player controller as well, sans the ‘if’ statement. I think the use of tags within Unity does a lot for reusability as well, as you don’t have to assign specific names, just apply the player tag to the player.
* How maintainable is your code? Is it easy to add more enemies, pickups, weapons, levels etc?
  + Its easy to add a new item as you can just create a class tied to a sprite. Currently the pickups are based off of a simple collision check, so just using the script for other items as a template makes item creation simple.
* Did you finish your game?
  + Yes! I was happy with how much I got done with my game. I was actually able to add an environmental hazard that wasn’t included in my proposal as well.
* How did the scope match up with the proposal and allotted time?
  + For my game, I think it did. I was able to finish my project and add what I needed to make it feel finished. The only think I would want more time for would be to possibly clean up the code a little more to allow room for expansion.
* If you didn't finish what's left todo?
  + Possibly adding better code optimization and cleaning.
* If you could redo the proposal what would you change?
  + I think I would possibly be a little more ambitious to challenge the complexity of the game. I think its in a great place, it just needs a little more depth.