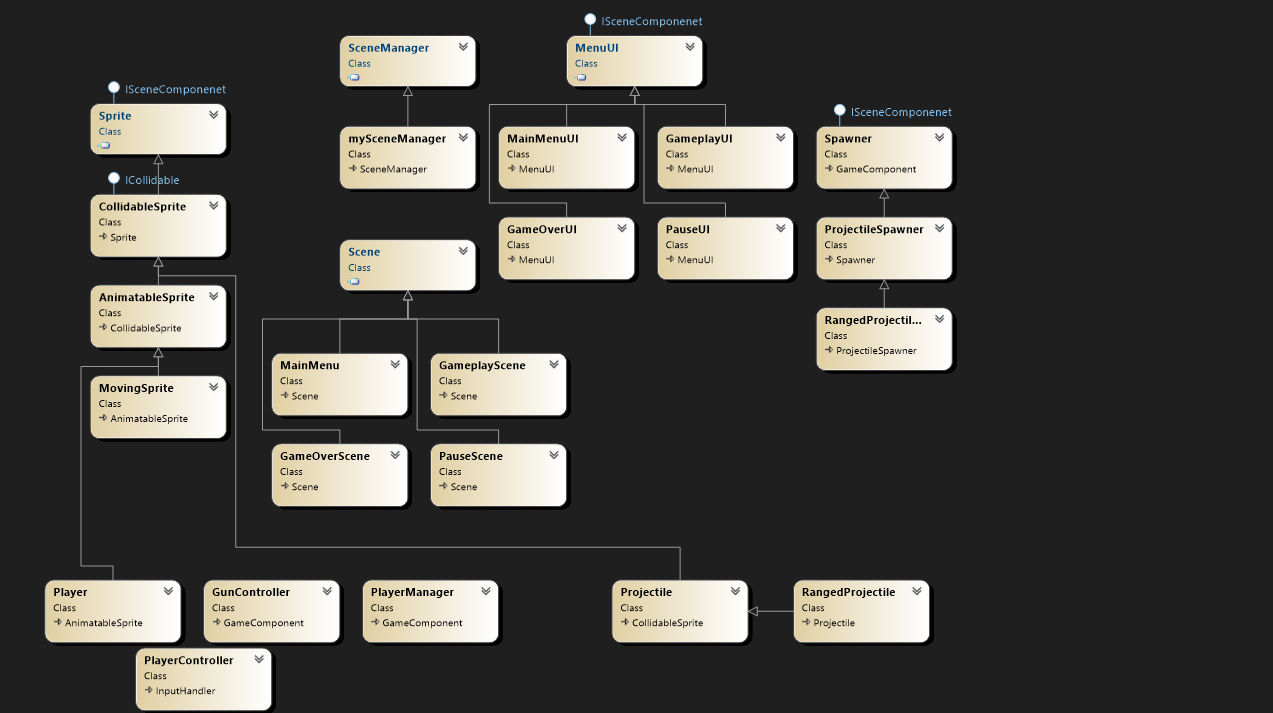
Name: Meteor Shower

Short Desc: A 2 player game where you and your partner are tied together, the goal is the survive the oncoming onslaught of meteors for as long as you can. Collect power ups to better your odds of survival.

Genre: Arcade Shooter

Credits: Me, Chat GPT for one or two methods, Jeff for helping me, and my math teacher Christopher Shaw for helping me with some equations.

Post-Mortem:



This UML shows a few of the hierarchies within my project. The main one is sprite, which has a few levels to it for variable utility. Collidable sprites use the observer pattern to add themselves to the collision manager. An animatable sprite allows for an animated texture to be played and overrides the draw for the original sprite. One important sprite is the player. The player manager has two players and keeps track of things like score and resetting the players, nothing else has a reference to players only to the player manager, so the players are encapsulated within the manager. This is a pattern I used with enemies and power ups as well. Each player has a player controller and gun controller. Both also encapsulate functionality and data. The controller has a input manager which reads input and translates it to bools. The gun controller has a bunch of projectile spawners which use the object pooling patten to keep the game running smoothly and not exhaust memory. One challenge I faced was trying to force patterns where they were not needed. I wrote about this in my devlog but I spent a long time with the gun systems trying to implement a decorator pattern and add lots of guns. I realized this was redundant and there was no reason I needed to have the guns stored elsewhere, the gun controller could just manage a list of all of them and set the active or inactive as needed. This was much cleaner, more efficient, and the task was accomplished with a lot less code.

The code base is probably 80% reusable and 20% game specific. A lot of the time building this game was sort of creating an engine from scratch as well. I wanted to challenge myself to do this, so I had a good understanding for making games in the future. I also don’t think I got as far as I had hoped, just because a lot of the time was spent understanding math behind how things worked. I also got far behind in the beginning trying to make one idea work (the rotation) which when I got working did not even feel how I expected. It was a great lesson but definitely burned me out and set the development back a ton. I’d say 80% of the code is specific to monogame as well, as a lot of it was essentially setting up a game engine. A lot of the code is already built into something like unity (and they definitely do it better lol.) I am happy to say that my code is very maintainable and its pretty super easy to add new items and scenes, without breaking anything. I got the game to where I had hoped. I did all of the hard work that I wanted to do and the core of the game seems to be there. If I were to take this project further it would be a lot of polish, adding new enemies, power ups, and weapons and so on. The core of the game is all there. Art and sound would need to be added to. I figured out pretty early on that I would not have time to do art and sound however so I am not disappointed by that. I had originally planned for this to be rhythm based, but I realized early on I would not have time for that either. It would be super easy to translate it into a rhythm game if desired however. I also spent time fixing a memory leak on my old game, and I’ve soaked that game for a long time so this game should not have any memory leaks or detrimental technical issues. One design issue I am simply to tired to fix at the time of writing this is that the timer resets when resuming game from pause menu. It would probably take me 10 minutes to fix it but I’m lazy and don’t know if showing time even fits. Need some more playtesting for this first.