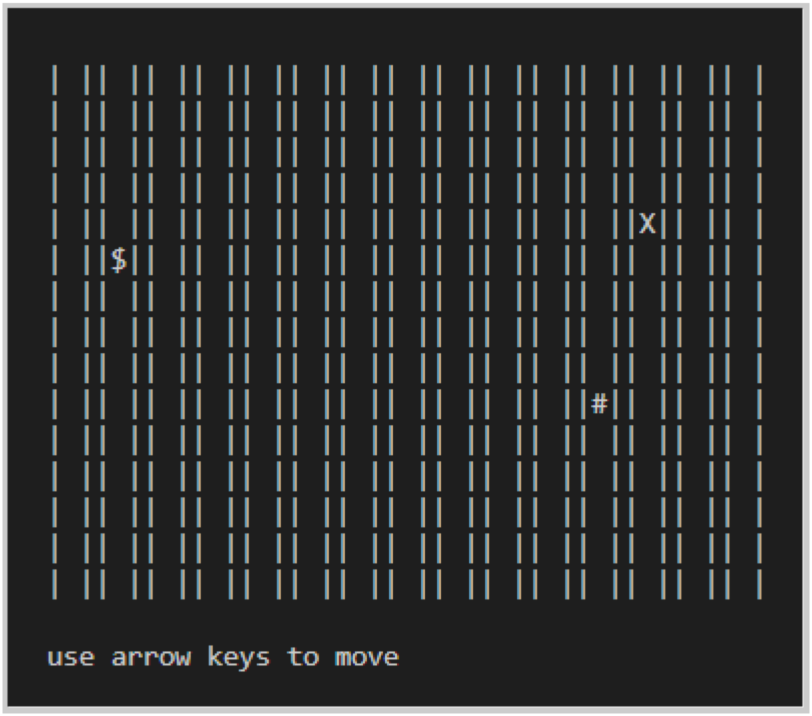
Custom project

Runner dash - CLI game

Game Overview:

I am making a command-line interface game in which the user can move his player using the keyboard’s arrow keys. This game will consist of a field of specified size, the player will spawn on a random location and has a chance to grab gems which will also spawn in the field. But be aware, the player is being chased down by enemies (one for now), and the enemies have a chance to move towards the player at every step the player takes. If enemies and your path collide, it’s game over. Your objective, just, for now, is to collect as many gems as possible and get the highest score possible, can you defy all odds to achieve success in this simple yet attention-grabbing game? Play now to see for yourself.



```
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || ||X|| || |
| ||$|| || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || ||#|| || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |
| || || || || || || || || || || || || || || |

use arrow keys to move
```

- X - player
- $ - gem
- # - enemy

Class Description:

My program firstly has a main class that starts the program's working which starts the working of the program and initializes any required objects.

Then there is the game class that does all the work of the game, it has many objects like players, enemies, and gems that are required for the working of the program. This class moves the player and the enemy and checks if the player lost.

After that comes the field class. This class does all the drawing of the game, the field you see being built in the game is made in this class, I have used a 2d array to make the game field. The coding is done such that we can just add a new block for each item we want to add in the game or even make a new method for multiplying certain already added elements (for example there will be multiple enemies in my next alteration of the game so this will help in that sector). For now, the size of the field is hardcoded but will later be customizable or built according to the level requirements.

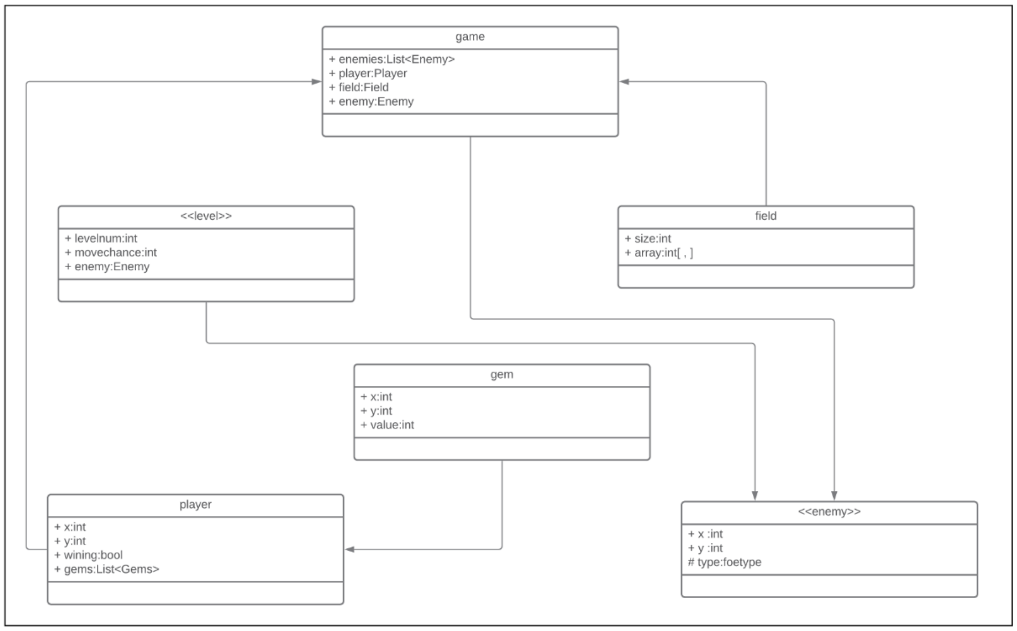
At last, for now, we have three classes for our three elements of the game.

The player class has the x and y coordinates of the player as well as a variable to add up the score. It also has a bool checker that checks if the player lost or not. The constructor of the player class takes the field as the parameter and sets random points for the player to spawn on the field.

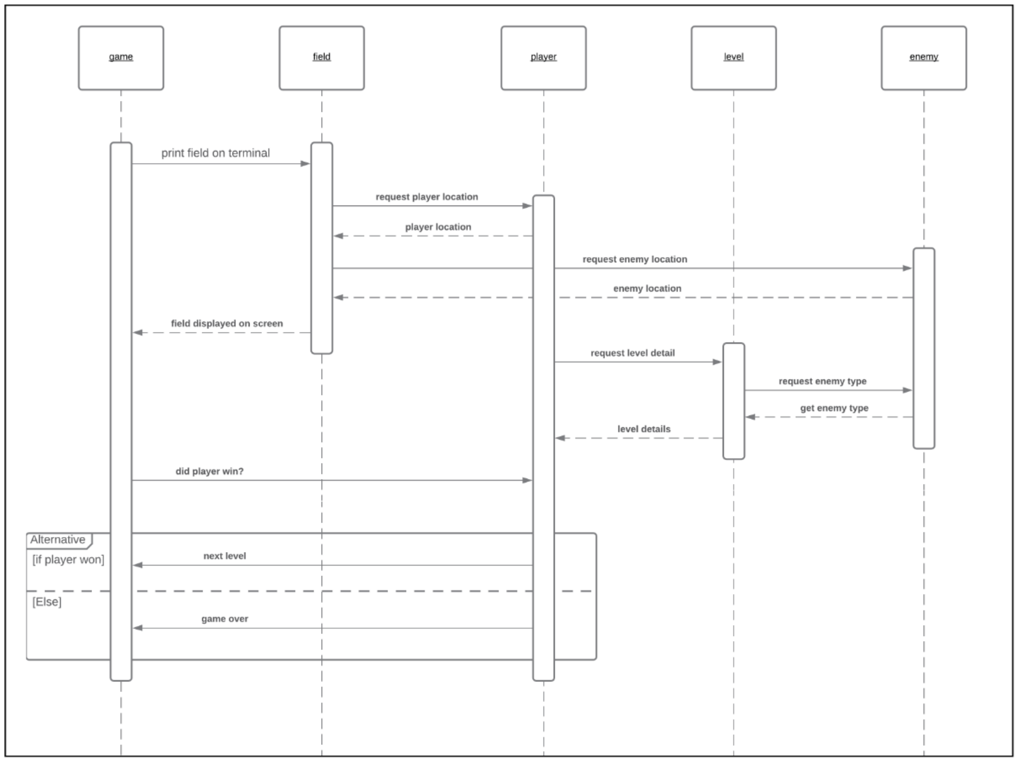
The enemy class consists of its x and y coordinates and a constructor that takes the field as the parameter and sets random points for the enemy to spawn. The gem class consists of its x and y coordinates and a value variable that can be used to set the value of the gems. It has a constructor that takes the field as the parameter and sets random points for the gems to spawn and also the value of the gem is initialized here.

Video link - https://www.youtube.com/watch?v=bsH5MnBXHlc

Uml class diagram



Uml sequence diagram


game
field
player
level
enemy
print field on terminal
request player location
player location
request enemy location
enemy location
field displayed on screen
request level detail
request enemy type
get enemy type
level details
did player win?
Alternative
[if player won]
next level
[Else]
game over