|  |
| --- |
| 1. Are your player, items and enemies the same class, different classes in the same family, or completely different classes?   The Player and Enemies are both the same class Sprite.  \*While making the game I changed it so the Player, Enemy and Item all inherit from the Sprite class. I did this because they all share some very similar variables and methods. By using inheritance it made comparing and storing the Player, Enemy and Item very easy because they all have the core logic provided by the Sprite class. |
| 1. What logic will you put into your Form class? What logic will you put into your Game Manager class?   The form will have a keydown method which calls a method from the Game Manager class to tell it what direction it is now facing as well as to move. It will also have a timer which during each timer tick calls a cycle method from the Game manager class.  \*I changed this slightly by adding in some code to the timer method which takes the values from the Player and enemy and displays it on the form using labels. |
| 1. What class(es) do you need to implement the dungeon? Briefly explain the job of each class, list the data members it must hold, and the methods it must expose. How do the Dungeon and the TileMap communicate?   C:\Users\Nick\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20161007_171003.jpg\*To implement the Dungeon Class I used a Room, TileMap and Cell class.  The Cell Class was used to hold the Index of the Tile as well as Boolean to tell if it was covered in fog. Cell had Getters and setters for the fog and TileIndex.  The Room class was used to randomly generate a room with a X and Y position as well as a Width and height. The Room then stored this information in a 2d array which could be added directly into another 2d array for the Dungeon. Room has Methods to add itself into a 2d array of the Dungeon. It also would add a portal into a room if told to.  The Tilemap Class has a 2d array of Cells, This 2d array was used to draw the dungeon by holding tiles at each point in the array. It has various method such as a Set and GetMapEntry which either gets or sets the cell at a specific row and column. It also has a DrawMap method which simply draws the contents of the 2d array to the screen, it will also draw fog if the tile is covered in fog. There is also a Calculate Position method that takes a X and Y position and returns what row and column that is.  The Dungeon Class has a PlaceHolder 2d array that gets edited multiple times before it passes all its points directly into the TileMap class. It then saves the instance of that TileMap so it may draw it later on. |
| 1. What data structure(s) do you need to hold collections of enemies and items?   I will use a SpriteList to hold the collections of enemies and items. |
| 1. Does the dungeon need pointers to its sprites? Why or why not?   No the dungeon does not need pointers to the sprites. This is because I want the Dungeon class to only be generating and holding the dungeon. |
| 1. Does the sprite class need a pointer to its dungeon? Why or why not?   Yes the Sprites do need a pointer to the dungeon this is so the sprites are able to figure out where they can be spawned and move about within the dungeon. |
| 1. What enum types (if any) do you need?   The Dungeon will require a enum of type int. This will be used to easily and correctly select the correct tile.  \*The Cell class has a enum to refer to the Tiles by a name instead of by a integer increasing the readability. |
| 1. Does the player sprite need access to the collection(s) of enemy sprites?   \*No the Player Sprite does not need access to the enemy Sprites instead the GameManager can pass the PlayerSprite into each of the enemy Sprites so that they can check if they overlap. |
| 1. What class is responsible for creating the collections of enemies and items?   The Game Manager class will be responsible for creating the collection of enemies and items. |
| 1. If you are using an FSM, what class calls the FSM methods of the sprites?   The Game manager class will call the FSM methods of the Sprites if I am using a FSM. |
| 1. At each game cycle, you need to perform collision detection between the player character and each enemy and item in the dungeon. What class or classes hold a method to compare the areas of two entities to check for collision? What is the function header of this method? What other classes are involved in the collision detection logic?   \*In the GameManager Class there is a method call to the CheckCollisions method from GameCycle. The CheckCollisions method then Tells the SpriteList to return a Sprite if it is overlapping with the Player. Within the SpriteList there is a method that checks each sprite to see if they are overlapping with the PlayerSprite and returns that sprite if it does. |
| 1. If you are implementing Line of Sight what algorithm will you use (i.e. room-based or field-of-vision)? What methods are needed, and which class holds each method?   I will be using room based line of sight because I do want the enemies to chase the player around the entire dungeon and down the corridors just when they are within the same room. |