|  |
| --- |
| 1. Are your player, items and enemies the same class, different classes in the same family, or completely different classes?   All players, items and enemies will be part of the same family with different classes. We plan on using inheritance and polymorphism to achieve this.    *Edit: Our sprite inheritance changed because we had to add extra classes like projectile and portal.*  *We also did not need the Combat class because it was doing nothing, so we removed it. Our new inheritance structure is shown below.* |
| 1. What logic will you put into your Form class? What logic will you put into your Game Manager class?   The form class will be the presentation layer. It contains everything relating to the user interface. This is the scoring system, player health, feedback, and buttons for New Game, Pause, and Exit. The form will also hold an instance of the GameManager class and the timer which runs the GameManager. It will create the graphics. It will handle the key down events and call the methods in the game manager. The display is in full screen and is dynamically generated depending on the screen size.  The GameManager class will contain the logic of the game. It will create instances of the other classes and call their methods. This class has the game cycle in it. |
| 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?   The classes that will be needed to implement the dungeon will be:   * Dungeon * Room   The job of the **GameManager** is to create an instance of the dungeon call the GenerateNewMap from the dungeon to populate the TileMap.  The job of the **Dungeon** class is to create and return a randomly generated 2d array of ints that represents the game map. The data members it will hold are an array of Room objects, max number of rooms, the size of the world, and the viewport offsets, and a random number generator. The methods this class will expose will be createNewRoom, CheckIfRoomOverlaps, createRooms (wrapper for CreateNewRoom and CheckIfRoomOverlaps, GenerateCorridors and GenerateMap.    The job of the **Room** class is to hold its starting row and starting column in the map, as well as its number of tiles wide and number of tiles high. We currently believe this class will not have any methods, however each of its data members will be properties. We may add to this later.  *Edit: We added addCorridorsToMap and addNewCorridor methods to implement corridor creation. These two methods were added to further break up code.* |
| 1. What data structure(s) do you need to hold collections of enemies and items?   Linked LIST |
| 1. Does the dungeon need pointers to its sprites? Why or why not?   The dungeon does will not need pointers its sprites because the sprites will receive a pointer to the TileMap which it can use to perform their tasks that need information about the world. |
| 1. Does the sprite class need a pointer to its dungeon? Why or why not   NO |
| 1. What enum types (if any) do you need?   Enums for states, directions, and room generation.  *Edit: We also added enums for Tiletypes, which was used by both game manager to fill Dungeon, and dungeon used them to know how to build dungeon.* |
| 1. Does the player sprite need access to the collection(s) of enemy sprites?   We do not believe that the player sprite will need access. Instead, we plan to give the enemy sprites access to the player sprite. This gives the enemy sprites the ability to make decisions/update their state based the properties of the player controlled sprite. |
| 1. What class is responsible for creating the collections of enemies and items?   The GameManager class will be responsible for the creation of all the sprites that will be present in each level. These will be created/deleted at the beginning/ending of each new level. This excludes the player controlled sprite as he will remain constant as each new level is created.  *Edit: The game manager created an empty list of projectiles, but the combat sprites added to the list.* |
| 1. If you are using an FSM, what class calls the FSM methods of the sprites?   Game Manager  *Edit: Linked lists call all the FSM sprites methods. But the game manager calls the linked list method.* |
| 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?   The sprites themselves contain the methods for collision detection. The spritelist calls these methods and passes in the player sprite.  *Edit: All sprites contain collision detection methods another sprite. This method is a boolean returning method call SpritesHaveCollided. If checks if the currents collision rectangle has collided with the passed in sprite (which is always the player sprite). The linked list is involved because it is the class the retrieves the player sprite from the game manager.* |
| 1. Describe the AI you are going to include.    1. Describe the behavior   Approach and Avoid.   * 1. Describe the implementation logic   Use trig to find angle towards target and use trig to calculate x and y velocity. This creates approach and the opposite of this is avoid.  *Edit: We used Approach and Avoid but made them a little bit more complex. We had to do this because we wanted sprites to walk either NSEW. We calculated the angle from the enemy sprite to the player sprite by used aTan. We then calculated the x and y velocities with Sin and Cos, and checked if they were positive or negative and based sprites movement on these values.* |
| 1. Describe the trigonometry you are going to include (if not already contained in the AI from item 12).   We plan to implement a range attacked that will create a projectile sprite. This sprite will use trigonometry to determine its velocity. This may include velocity.  *Edit: We used the same methods as above to calculate angle towards target and find x and y velocities to implement projectile movement. But we did not set it to a fixed direction. We also implemented gravity so the projectiles look more realistic. The projectiles are destroyed and removed from the list when they collide with the walls or player sprite.* |
| 1. Describe, in detail, the logic of your battle algorithm and computations.   We plan having sprites shoot out other sprites and if there is a collision the shot is successful.  We will also implement simple battle. This involves two sprites colliding, if one is in the attack state then other the will lose health.  *Edit: We added an attack animation to our player sprite. An enemy will only lose health when the player is attacking (and performing the animation). The enemy sprites do not have an attack animation, but if they are colliding with the player sprite there is a random chance of them attacking and taking health off the player. Furthermore, the enemies have a random chance to throw projectile if the player is inside their search zone.*  *The combats sprites and the player sprite each have their own attack damage. The combat sprites damage is increased each time the player enters a new dungeon. The amount of health lost with each attack is the amount of attack damage the sprite has.* |
| 1. Sketch the screen layout -- with -- controls that you will use to provide feedback during battle. |