Designs #3
CS 202: Programming Systems

Alves Silva, Otavio Augusto
PSU ID: 902840168

Design #3

The program #3 will be a representation of the game Halo. As a FPS game, we will need some classes to represent and build the environment. For example: player, battlefield, weapons, list of weapons, ammunition, history line and etc. With these classes and can start thinking about the creation of the game.

1. Class player

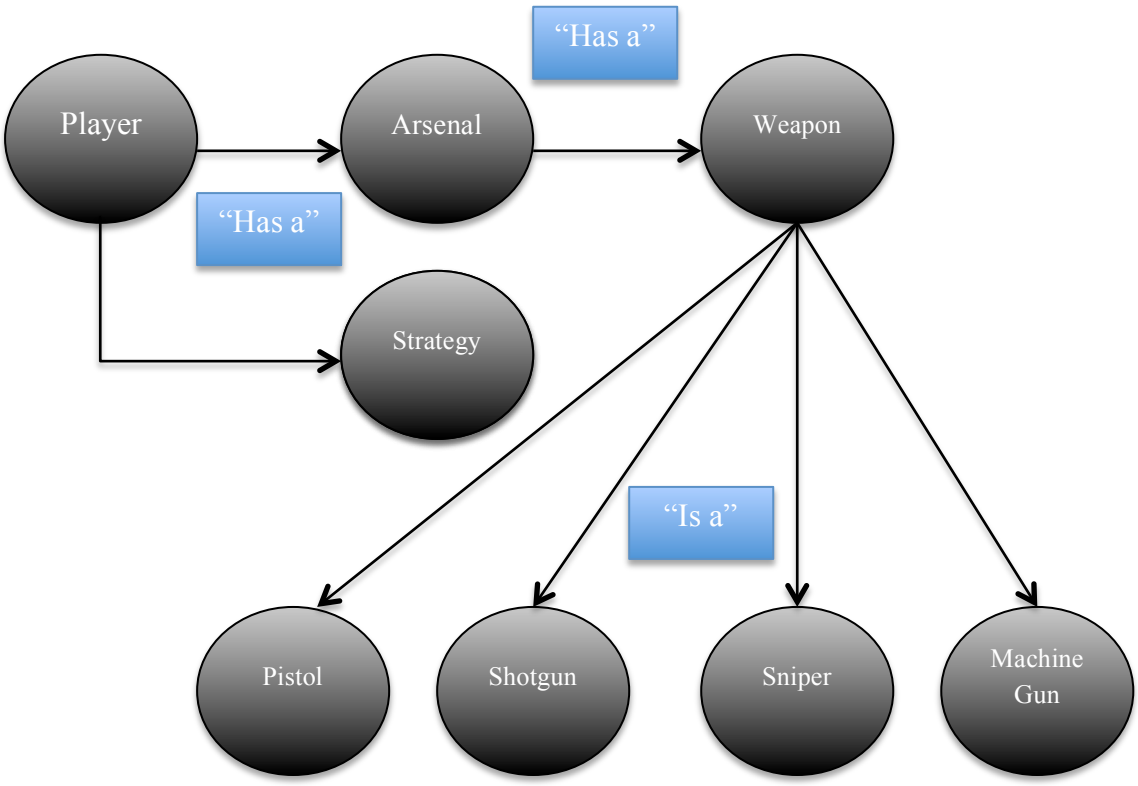
All game needs a player. Therefore, the first class that will be created is the class player. In the game the player will do some actions like to walk through the battlefield, take a weapon, discard a weapon, fire and reload the weapon. The player doesn't have just one weapon, so an arsenal could resolve the issue of hold just one weapon. Each of these nouns can be used and so be a class in our program.

1. **Player**: This class will hold the player's name. It will have the following functions and data members.
    a. Functions
        i. Executes a action
            1. Walk through the battlefield
            2. Fire an enemy
            3. Reload the weapon
            4. Change weapon
            5. Discard a weapon
            6. Check the available weapons
    b. Data members
        i. Name
        ii. Description
        iii. Arsenal
        iv. Strategy
2. **Weapons**: This class will represent a weapon in the real world. It will have the following representation
    a. Functions
        i. Randomly algorithm to create a weapon
        ii. Display the available ammunition
        iii. Display the available ammunition chest
        iv. Reload the chest

b. Data members
    i. Name
    ii. Ammunition
    iii. Damage

3. **Arsenal**: This class will hold a list of weapons available for the player. It will have the following functions and data members.
    a. Functions
        i. Display actual weapon
        ii. Next weapon
        iii. Previous weapon
    b. Data members
        i. List of the weapons
4. **Strategy**: This class will hold the past movements of the player. It will have the following functions and data members.
    a. Functions
        i. Insert a new history
        ii. Shows the actual history
    b. Data members
        i. List of wins and loses through the history

After set the classes, which have a relationship with the player, we have to explain the relationship between those classes. The following UML diagram can show a brief idea about the design of the player class.



2. Class Battlefield

The battlefield will be story, which the player will play, created randomly each time that the player starts a new game. It has different amount of enemies and Spartans, different types of weapons. This class will basically represented by balanced BST. So right now, we have some nouns that we have to create for build the battlefield class. The following descriptions of each class explain each responsibility and function of the classes that will be related with the battlefield class.

1. **Random algorithm**: This class will have a random algorithm to create and build the history each the player will play. Therefore, it will create the balanced BST, where each path on the battlefield will have arsenals and enemies and different firefight, which the player can find during the game. It will have the following functions and data members.
    a. Functions
        i. Create the battlefield
        ii. Set the history for each path
        iii. Destroy the battlefield
    b. Data Members
        i. Number of paths
2. **History**: This class will represent a prize for the player. Therefore, it will give points for the player when the player passes through this path. It will have the following functions and data members.

    a. Functions

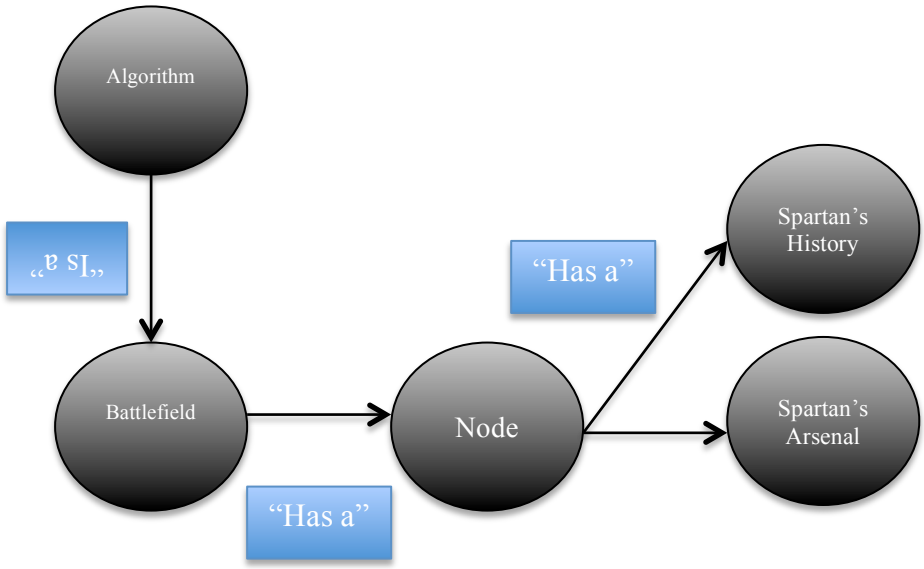
        i. Set the history action
        ii. Implement the action in the player
    b. Data Members
        i. History fact
3. **Enemies**: This class will represent an enemy that the player can find during the game. It will have the following functions.
    a. Functions
        i. Shot the player
    b. Data members
        i. Amount of damage

4. **Node**: This class will represent the paths through the balanced BST. Each node will have a DLL of the Spartan history and a particular Spartan arsenal that will be generated by the algorithm. Therefore, it will realize the history action in the game depending of the player action.
    a. Functions
        i. Go to the left

ii. Go to the right
iii. Connect to the left
iv. Connect to the right
v. Connect to the left
vi. Implement history and arsenal
vii. Set the action in the player history

b. Data Members
i. Left
ii. Right
iii. Spartan History
iv. Spartan Arsenal

After set the classes, we need to explain the relationship between those classes. The following UML diagram can show a brief idea about the design of the maze class.



This design can resolve the assigment, but somethings may change over the develop time.