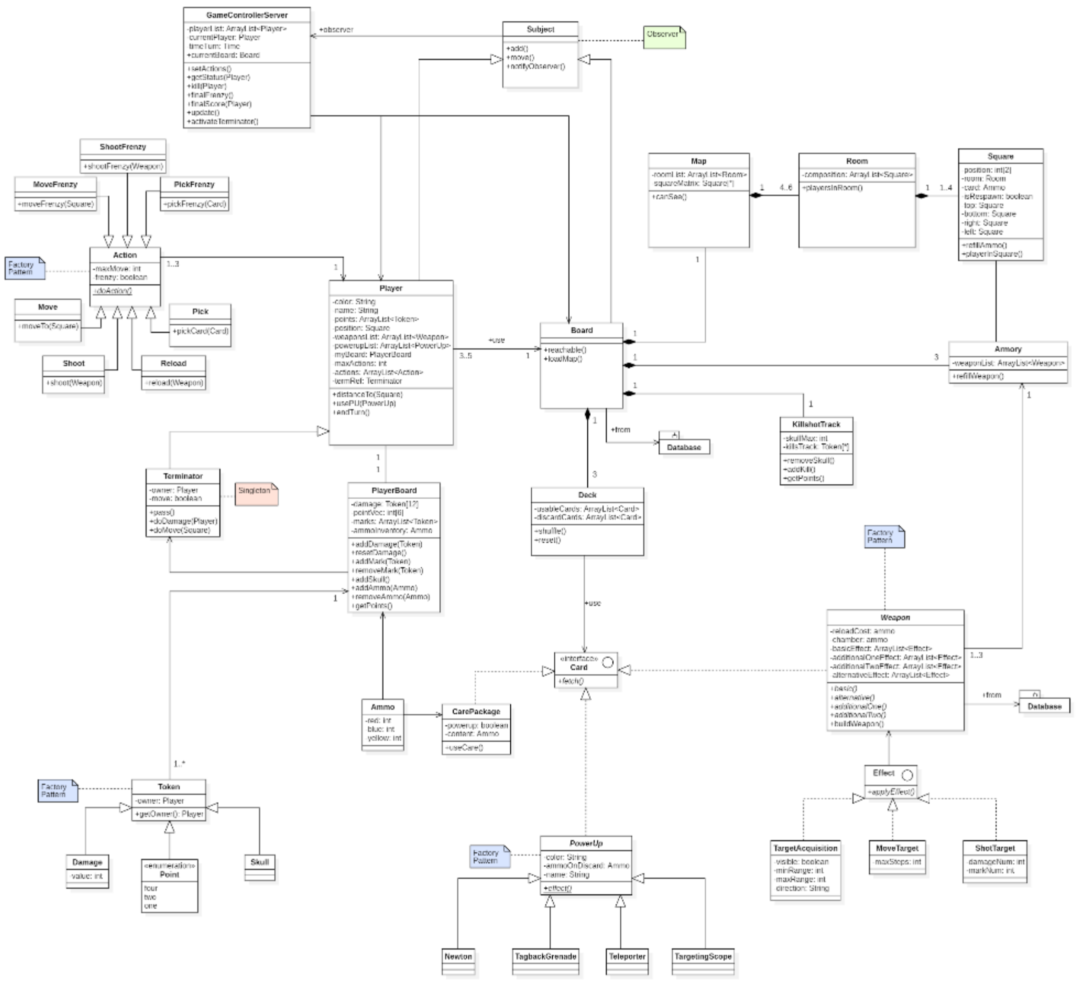
UML Documentation

Group 14: Barzasi - Bellacoscia - Cai



GameControllerServer

GameControllerServer manages the dynamics of the match, processes the turns and manages time, turnTime is the time a player has left to complete his turn.

The GameControllerServer dictates the set of moves a Player can perform, receives instructions from the **GameControllerclient**.

The status of the game is updated to the **GameControllerServer**, **Player** and **Board** using a subject according to the observer design pattern, allso initiates final frenzy and calculates the final score.

Player

Player represents a player in the game, contains its position and his available assets and resources.

ammoInventory contains available ammunition and is represented with an **Ammo** class which contains three fields of integers each representing one ammunition type (red, blue or yellow)

Action

An action represents an action a player performs during the turn, frenzy is a Boolean which when true activates Final Frenzy actions. Actions are instantiated by a factory class which takes player inputs and generates the corresponding action. The same design pattern is applied to the generation of **Tokens** and **Weapons**.[2][3]

PlayerBoard

Playerboard contains information about the damage received by the player, his marks and the points given upon his death.

Token[2]

Token represents a token of the game which can be a Damage, Mark, Skull or an amount Point (1,2 or 4), each token is assigned to a player's board when instantiated.

Room

Room is the same concept as a room in the game, contains a set of cells (**Square**) which compose the room.

playersInRoom returns a list of players in the room.

Square

Square represents a single cell of the map, position is a set of 2 integers which represent its coordinate in the map.

card contains information about the resource card on the cell and refers to a CarePackage[1] object.

isRespawn() specifies if the cell contains a respawning point in which case it will not have resources but will have an armory.

refillAmmo() refills the resources at the end of each turn if grabbed by a player.

Armory

Armory contains the set of weapons up for grab in a cell.

refillWeapon() refills the taken weapons if the weapon deck has not run out.

CarePackage [1]

CarePackage is a set of three resource units (either three ammunition or two ammunition and one powerup)

content is expressed with an ammo data type, upon grabbing the values are summed to the player's ammo inventory with a maximum cap of three.

hasPowerup is a Boolean specifying whether the care package contains a power up, if yes, the sum of the fields of content will be 2 instead of 3.

Deck

The deck keeps track of the **Cards** using an array, the generation of power ups and ammunition are not randomised but pre-determined so if an advanced player decides to count cards the game allows him to do so unlike most casinos.

Card

Card is an interface representing a single card of any type.

Weapon[3]

Weapons are implemented using the singleton design pattern because each weapon is unique and two players cannot have the same weapon, reloadCost is the full cost of reloading the weapon while the chamber attribute specifies the chambered ammunition when picked up, pick up cost ill be calculated as (reloadCost – chamber).

Effect

Effects are managed by three main Classes: **TargetAcquisition**, **MoveTarget** and **ShootTarget**.

TargetAcquisition verifies whether or not a player can be shot at by its weapon, since most weapons either shoot a target they either can or cannot see at a certain range (a maxRange or a minRange) condition or into a single direction, only 4 parameters are needed to define their function, exceptions will be handled singularly.

Move defines how a target will be moved in case the effect allows so.

ShootTarget is the damage the weapon deals upon shooting and the number or Marks given to the target.