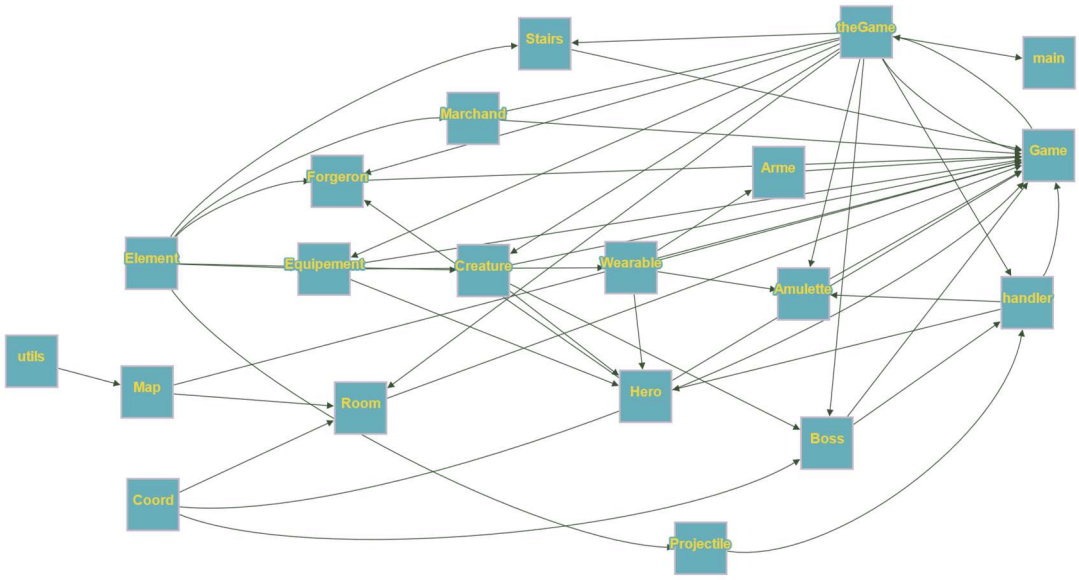
RAPPORT ROGUE-US :

Members: Clement Andy / Martini Alexis / Pantani Timothé / Dufraigne Titouan

Welcome to our game, ROGUE-US! Before you start, please read the READ ME. Link to the flowchart: http://graphonline.ru/fr/?graph=ICkvfVBkdeEHqwtvZZcst



Code Flowchart

Classes and their methods:

Amulet :

- regeneration()
- coupCrit()

Boss :

- spawnArmy()
- tpBoss()

Coord :

- distance()
- direction()

Creature :

- description()
- meet()

Element :

- description()
- meet()

Equipment :

- meet()
- use()

Blacksmith :

- meet()

Game :

- buildFloor()
- buildBoss()
- buildStairs()
- addMessage()
- readMessages()
- randElement()
- randEquipement()
- randMonster()
- select()
- pause()
- drop()
- selectPerso()
- restart()
- play()

Handler :

- heal()
- teleport()
- tir()
- boost()
- nerd()
- revive()
- fullBoost()
- amasser()

Hero :

- description()
- fullDescription()
- checkEquipement()
- take()
- use()
- unEquip()
- inventSize()
- shot()

- throw()

Map :

- addRoom()
- findRoom()
- intersectNone()
- dig()
- corridor()
- reach()
- reachAllRooms()
- randRoom()
- generateRooms()
- checkElement()
- put()
- get()
- pos()
- rm()
- move()
- moveAllMonsters()
- vision()

Merchand :

- meet()

Room :

- intersect()
- center()
- randCoord()
- randEmptyCoord()
- decorate()

Stairs :

- meet()

TheGame :

- theGame()

utils :

- sign()

Wearable :

Modifications Present in the Suggested Elements List:

Gameplay

Gameplay Element III (Limited Inventory):

- Hero.size: Instance argument that stores the inventory size.
- Hero.inventorySize(): Checks if the inventory is full. If the number of items in the inventory is less than the limit, returns True.
- Hero.take(): (Modification). Does not take an item if the inventory is full.
- Hero.description(): (Modification). Adds an item counter.

Gameplay Element IV (Semi-Intelligent Movement):

- Map.moveAllMonsters(): Creatures can avoid obstacles on their path to reach the hero.

Gameplay Element VI (Visibility Cloud+):

- Hero.vision: Variable corresponding to the hero's visibility.
- Map.vision(): Method that copies the map and returns its equivalent with fog.
- Map.fog(#): Fog hiding the map

Objet

Object Element II (Weapons and Management):

- Wearable.__init__(): Manages the placement of equipment.
- Creation of the Weapon class (inherits from Wearable): Allows differentiating consumables from weapons.
- Hero.body: (Instance argument) List representing the hero's body and managing equipped items. Each index represents a body part.
- Hero.use(): (Modification). Distinguishes between consumable items and equipable items. Consumables are used, and equipment is added to the body.
- Hero.unEquip(): Allows unequipping an item from the body. The item returns to the inventory.
- Handler.boost(): Increases the hero's stats (when equipping an item).
- Handler.nerf(): Reduces the hero's stats (when unequipping an item, for example).
-

Object Element III (Throwable Weapons) :

- Hero.shoot(): Checks if the player has a ranged weapon equipped. If so, it shoots.
- Hero.throw(): Asks the user in which direction they want to shoot.
- Handler.shoot(): Used to shoot an arrow in a given direction. Can be used by the hero or creatures and can give effects like poison or freeze to the target (with a probability).
- Creation of the Projectile class, which inherits from Element. It can have range, strength, and give effects. The projectile uses Map.move() to move until it reaches its range or a target, then disappears.

Object Element IV (Armor):

- Creature.defense: Used to mitigate damage if the player has a defense greater than 0.

Object Element V (Amulet):

- Hero.__init__(): Adds a movement counter for amulets.
- Creation of the Amulet class.
- Amulet.regeneration(): Regenerates the hero's health every n movements.
- Amulet.criticalHit(): Every X turns, significantly increases the hero's strength for 1 turn to deal a critical hit.
- Game.play(): (Modification). With each movement, increments the amulet counter to apply different effects.

Rooms

Room Element III (Shop):

- Creation of the Merchant class (inherits from Element), allowing the purchase of any object in exchange for gold.
- Hero.money: Instance argument to store gold for purchasing items from the merchant.

Monsters

Monster Element III (Archers):

- Map.moveAllMonsters(): Modification. If the creature attacks at range and the hero is within range, it shoots in the hero's direction instead of moving.
- Uses the Handler.shoot() function, explained in Object Element III (Throwable Weapons).

Modifications Added by Our Imagination:

Gameplay

Gameplay Element (Hero Selection):

- Game.selectCharacter(): Asks the player which hero they want to choose. All have different characteristics, which can be read by pressing "i".

Gameplay Element (Score):

- Game.play(), Map.move(): Modification to display a score at the end of the game. It is calculated based on the number of floors reached and the number of creatures defeated.

Objet

Object Element (Philosopher's Stone):

- Creature.life: Used to "resurrect" the creature if its life reaches 0 by giving it back all its life if the creature has more than one life.

- Handler.revive(): Method that gives 1 life to the creature.

Object Element (Effects (Poison/Freeze)):

- Map.moveAllMonsters(): Some weapons can poison or prevent the enemy from moving. This occurs with a probability and for a defined number of turns. The freeze and poison effects on creatures occur in moveAllMonsters. The hero's poison effect occurs in Game.play().

Object Element (Full Boost):

- Handler.fullBoost(): Method that increases HP and max HP by 5, attack by 2, and defense by 2.

Rooms

Room Element (Blacksmith):

- Creation of the Blacksmith class, which inherits from the Element class, allowing the improvement of a weapon or armor that has never been improved by increasing its statistic by 1 and giving 1 steel.
- Hero.steel: Stores the number of steel to later improve a weapon or armor.

Monsters

Monster Element (Boss):

- Creation of the Boss class: Allows differentiating classic monsters from bosses.
- Boss.spawnArmy(): Method that allows the boss to summon a random creature every n turns.
- Boss.teleportBoss(): Allows the boss to teleport.
- Map.move(): (Modification). Adds functionality. If a certain boss takes damage (here Thanos), it teleports.
- Game.boss: List containing bosses and their types.
- Game.buildBoss(): Adds more or less HP and strength to the boss depending on the floor.
- Game.buildFloor(): (Modification). Adds the boss instead of the stairs (you must kill the boss to access the stairs).
- Game.buildStairs(): (Modification). Adds stairs to the center of the last room or nearby points.
- Map.play(): (Modification). Every time the hero moves n times, a certain boss (here Joker) summons its minions.

Work Distribution:

Timothé:

- Project leader.
- Optimization of various lines of code.

- Visibility cloud +.
- Debugger, compiler.
- Pause menu, ability to exit the interface with an exit key.
- Optimization of effect assignment and blacksmith.
- Monster drops.

Alexis:

- Poison and freeze effects for the hero and creatures.
- Projectile.
- Objects using projectiles (bow, poisoned bow, freeze bow).
- Creatures that give effects to the hero (archer, poisoned archer, ice mage).
- Creatures that shoot at range (dragon).
- Score.
- Creature movement.
- Character selection.
- Merchant optimization.

Andy:

- Creation of amulets and their abilities.
- Creation of weapons.
- Equipment management (body, use, unEquip, as well as boost and nerf functions).
- Limited inventory.
- Improvement of the heal function.
- Creation of bosses, their implementation, and their abilities.

Titouan:

- Creation of armor (gold, silver, and bronze) using the defense statistic.
- Blacksmith improving a weapon or armor.
- Merchant selling an item.
- Some objects: FullBoost, philosopher's stone.