(lien pour le diagramme de classe du début de sprint)

[http://www.plantuml.com/plantuml/svg/jLdVJkCw4x\_FKrXmazMsQ3zz5oiXXBP0ePLQMhLXWRDQhKngccZJEB9TeEBmxYV-quapIK\_5YdDJuj-Cn-Fnp7Z6V9EA2hLUPK51uzzqoSZ1zvGz7v3NWCGPbPBeuIqNsK93v3EP5o9Lx8ogE27omK0JeHA-5BH8qlYShfWWiVwbXWZpmqzcB5Y-O29yp3XL5h7Ibqm1svk\_UBHMVBksqXfA\_Vidke2udQeyyNH1PCAV6rgCMRx-62r0t3krk4mpTfvmBfbmYiI2KSL6x8bbOPehakalC-KjgCvpnHSiCmSAK9nGGMF5X849VP1AZzr2vrkg8P93ecKdHVLDbIebLUE2oxkc6VRO0Rj6cobQdSDRzRuQBLB1OfNot48Fd6UCvaGbWg-NoItDCakmMX9f\_W04YZ\_etu30AIm4VGxhE\_laaLVyYStIPQ92Z8xOOnECUD70phXIV5M3rLhaOMrJFdN1zBbBDYWi3rtH9VCP95DtXlI2q7k3RXoglvTCJSl32nlGlOF0aAcCAbCY\_a61W9vnaDgUqjnfFJLA98ncAj4CfQiIFO-9HnepCYvDicAvShxNQkP3P6O0iPaFiPa1RZSpw6C3OSnp2M5Wz-5y\_vndMIhB6Q7rpsVzYsPV9Lp2DbwbeYgDtuo5vjFPCEhFXt\_rZM3BQK-iOnWsIDGqenlGeoxhV1hrheVZNZISN6DoRQdRuTKWgd7yH1o3BKUtZIEQJ7VIpYPHD1dl98\_wbxlbpgfLZGMixrJ0fW8-EqCWK-rsik8gVM8vYeABxMc3V-uzLymsB\_45Pl5JHfAgHjR0GixuyxzwNoHAPV2POz-nNAs-y-Ob\_8m-R4H9JTeJFjikr8LubpnB5rLwVQa-DovhgbBWagXDmInGPOfUhD8dgfZHq4XlEx7NhHPlpYfk5hRh\_bJiL5ao5PNQe5o4azjzYFnXgwt1ty3y\_LMXDixyTe0sWTBZN1iCPE\_UaB\_6jFpHv\_DQkJiLelJPEmpSMxbkGhYzTJjm4tRysVxWVg-cDYgOL6bEjH\_YVRvZGv6W1Um6BYSZmj1geZiUov-R4-XGjemu5GicMlm5AykeN0lc3Xd67tJExp3RBOL8DzyVxsbZkanZjuGTVFGkzvfxlD6NAOJK7wq5KNpESyLo5RvBWmlEuHBzJxSwsgRRMka0QAZhxny\_V\_maYZwCRD4EgTBi3gTCHBzcBqsU3e6RxPB6YeiDgX\_tQW-jHjNlTGaq5W58MDALoPPx97f5xe097kvM3t6xkgj-EzfTieojBBVD6rL1u01BL2LVmOLghHTWhUSqTmmU62Si\_ZtYaW79TvUOvhf23QiUATvZTwXQzC61p3srlYDckjiBiH1ZJ0kEKgbEdBbEDSrzIty9lqsiO5E2U\_P9YiOK96pADTzZcY\_CTMWSq3yv6zz6MEpXUUciEl8CBmGm8B1LkcYCTSjB7ZrKqYTd3uY5orH8TKiRmO1j-XPuNOQjALr9VRYh0eJODxDHQ8zA1Q4JHwXJ4L1dJnWRLVFrQWGDEswzA-dxJzb9XxajqlM3WEjAw8gHLtJIzMXEPM7Tpdz5Y7ckTAd9HFKsP1HZkTEpIyPhcSPaLVn\_pjVvma1dDcwCcvkJcWecfLuC9eVm350FjTziSy6VJN0q23T5AwpBluN9hdMIukiusGhhaDfF2wULtifd9GcwN9OBxPQZIOs7c0kggCiGdWqZzb8QXpa2EJaXuRnpU7dvzUkNBv\_8FCsN6LCy9wUdpie3AfCnBKvirTjnyqz9gjEVDIsIkMHgc4j5yzWUT1SnkDDN5ZJiMbPeLt0U511M-bGprrbVWlx4jeaU0BsXXzc6q4TC5uWatSokhJG4WaI4MwaOjPBD5d08xzZ2eDzKvdPu4GqGr1kDwiZLR79pVL77BeUpVXsvw8-YNXsQzkRptXNY6\_UkXkSjqA3V6s5TfgFUVN-6j1bCvjWUimdM-Vhir\_NmUxztSuUCCFrbXJelnfcfsptuty7sKZec2HEyZUTBv-Yean\_OfnJ3kR9VCe2mTxBizx5DkG6y\_5gYILl7fl88u3csIVBVI9jKct6010gKzAVjTxK9SdJqzocyHmKefem4X1qJlXRRI0libgqOk700xNPB5DOAx6EH0Hs7XT1iQsyysRuKm6I0q6JRqTgvBCi2rqnwY6dVgY4qI5tp-0QuQLSgDjjRL6DNw7k8ARohONKlTlNxNUabHqVvWQ6LDepBF\_ga0d0Ex6X0SHERpkNFHPUpPilgI3jt3KMkhTK41CkmJXVe7pVUUcpmZUKB\_S-M8FW7](about:blank)

@startuml

package "View" {

class ViewWorld {

+ SpriteBatch sb

+ OrthographicCamer camera

+ ViewWorld()

+ render(float)

+ World getWorld()

}

class ViewTuto {

+ Stage stage

+ SpriteBatch sb

+ OrthographicCamer camera

+ ViewTuto()

+ render(float)

+ void show()

}

class ViewMenu {

+ Stage stage

+ SpriteBatch sb

+ OrthographicCamer camera

+ ViewMenu()

+ render(float)

+ void show()

}

class FileChooser {

+ createLevel(int level)

}

}

package "model" {

package "character" {

abstract class Character {

# int hp

# int attack

# int posX+ SpriteBatch sb

+ OrthographicCamer camera

+ ViewWorld()

+ render(float)

+ World getWorld()

# int posY

# int direction

# boolean throughWall

# Sprite sprite

# SpriteBatch batch

+ void draw(SpriteBatch sb)

+ void moveRight()

+ void moveLeft()

+ void moveTop()

+ void moveBottom()

+ void turn(int direction)

+ void decreaseHp(int damage)

+ void setX(int x)

+ void setY(int y)

+ int getPosX()

+ int getPosY()

+ boolean isThroughWall()

+ int getHp()

+ int getAttack()

+ void increaseHP(int health)

}

interface Movement {

# void moveRight()

# void moveLeft()

# void moveTop()

# void moveBottom()

# void turn(int direction)

}

class Monster {

+ void attackCollision(World world)

}

class Ghost {

# {static} int SPRITESIZE

+ Ghost(int x, int y)

}

class Player {

+ {static} int CPTANIMATION

+ {static} int SPRITESIZEWIGHT

+ {static] int SPRITESIZEHIGHT

- {static] int SPRITETOP

- {static] int SPRITEBOTTOM

- {static] int SPRITELEFT

- {static] int SPRITERIGHT

- int xAttack

- int yAttack

+ Player(int x, int y)

+ void increaseHP(int hpGiven)

+ void setDirection()

+ void attackCollision(World world)

+ void attackSword(World world)

+ void drawhit(SpriteBatch sb)

+ void attackBow(int direction)

}

class Troll {

+ {static} SPRITESIZE

+ Troll(int x, int y)

}

}

package "labyrinth" {

abstract class Cell {

- boolean solid

# int x

# int y

# String type

# boolean isActivate

+ Cell(int x, int y)

+ {abstract} void activate(World world)

+ void draw(SpriteBatch sb)

+ String getType()

+ int getX()

+ int getY()

+ void setActivate()

}

class Empty {

+ Empty(int x, int y)

+ boolean isSolid()

+ void activate(World world)

}

class Passage {

- int arriveX

- int arriveY

+ Passage(int posX, int posY, int arriveX, int arriveY)

+ boolean isSolid()

+ void setDestination(int x, int y)

+ void activate(World world)

}

class Trap {

- int damage

+ Trap(int x, int y, int damage)

+ boolean isSolid()

+ void activate(World world)

+ void CharacterLoarderdraw(SpriteBatch sb)

}

class Treasure {

+ Treasure(int x, int y)

+ boolean isSolid()

+ void activate(World world)

+ void activate(World world)

}

class Wall {

+ Wall(int x, int y)

+ boolean isSolid()

+ void activate(World world)

}

class Magic {

+ Magic(int x, int y)

+ boolean isSolid()

+ void activate(World world)

+ void draw(SpriteBatch sb)

}

class Mystery {

+ boolean isSolid()

+ void activate(World world)

+ int getContent()

+ void draw(SpriteBatch sb)

}

class Door {

+ boolean isSolid()

+ void activate(World world)

+ void draw(SpriteBatch sb)

}

class Labyrinth {

+ void Labyrinth(char[][] tabLevel)

+ Cell getCell(int x, int y)

+ Cell getNextCell(int x, int y, DirectionFactory direction)

+ void draw(SpriteBatch sb)

}

}

class World {

- int level

+ void game()

+ void draw(SpriteBatch sb)

+ void moveHero(DirectionFactory direction)

+ void moveMonster(int direction, int elem)

+ damagePlayer(int damage)

+ teleportPlayer(int x, int y)

+ void winPlayer()

+ void checkLosePlayer()

+ boolean canMove(Character c, DirectionFactory direction)

+ Player getPlayer()

+ void turn(int direction)

+ void loose()

+ Player getHero()

+ List<Monster> getMonsters()

+ void createLevel()

+ void healPlayer(int hp)

+ void createMonsters()

+ Point findEmptyCell()

+ void mystery(int c)

+ void nextDoor()

+ int getDoor()

+ void load()

+ void load(File f)

+ void save()

+ Cell getFirstWall(int x, int y, int direction)

}

interface Attacker {

# void attackCollision(Wold world)

}

class LabyrinthLoader {

+ LabyrinthLoader()

+ Labyrinth createLabyrinth(int numLevel)

}

class CharacterLoader {

+ List<Monster> getMonsters()

+ createCharacter(int numLevel)

}

}

class Game {

+ void create()

+ void dispose()

}

package "controller" {

class Listener {

- Music mp3Sound

- Boolean turnFirstPress

+ Listener(ViewWorld)

+ boolean keyDown(int)

+ boolean keyUp(int)

+ boolean keyTyped(char)

+ boolean touchDown(int, int, int, int)

+ boolean touchUp(int, int, int, int)

+ touchDragged(int, int, int)

+ boolean(int)

}

}

package "DataFactory" {

class TextureFactory << (S,#FF7700) Singleton >> {

- HashMap<String,Texture> images

+ TextureFactory getInstance()

- TextureFactory()

+ texture getImage(String)

}

enum DirectionFactory {

+ {static} TOP

+ {static} BOTTOM

+ {static} LEFT

+ {static} RIGHT

+ {static} TURNTOP

+ {static} TURNBOTTOM

+ {static} TURNLEFT

+ {static} TURNRIGHT

}

class LabyrinthFactory {

+ {static} HEIGHT

+ {static} WIDTH

+ {static} WALL

+ {static} GROUND

+ {static} FIRE

+ {static} DELTA

+ {static} PASSAGE

+ {static} MAGIC

+ {static} MAGICHEALTH

+ {static} PLAYER

+ {static} GHOST

+ {static} TROLL

+ {static} NB\_NIVEAUX

+ {static} HP\_PLAYER

}

}

CharacterLoader --"1" Player : hero

CharacterLoader --"0,..,n" Monster : monsters

ViewWorld --"1" World : world

ViewTuto --"1" Game : game

ViewMenu --"1" Game : game

FileChooser --"1" Game : game

Listener --"1" ViewWorld : viewWorld

Character ..|> Movement

TextureFactory -"1" TextureFactory : ourInstance

Monster ..|> Attacker

Monster --|> Character

Ghost --|> Monster

Ghost ..|> Attacker

Player --|> Character

Troll --|> Monster

Troll ..|> Attacker

Empty --|> Cell

Passage --|> Cell

Trap --|> Cell

Treasure -Right-|> Cell

Wall --|> Cell

Magic --|> Cell

Mystery --|> Cell

Door --|> Cell

Labyrinth -Left-"0..n" Cell : cellList

World --"1" Player : hero

World --"0..n" Monster : monsters

World --"1" LabyrinthLoader : labyrinthLoader

World --"1" Labyrinth

World --"1" CharacterLoader : characterLoader

World --"1" Game

Game --"1" ViewMenu

@enduml