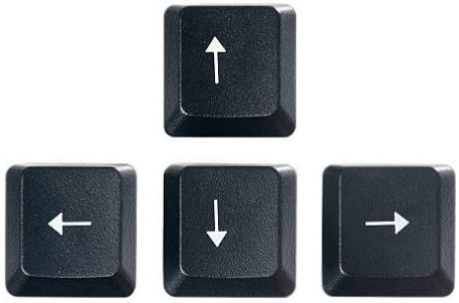

Jeu 2D
HP:1/5
Damage Taken: 24pts
Status: You Win!!

Main Objective of the Game

To outrun and avoid all **Ibol** monsters that are rooming the game to finally arrive at the safe zone (last Map) without taking any damage. All **ibols** have **AI** system for their movement.

Controls



We decided to go with the classic arrow keys to navigate through the game.

To program the Game, we imported classes that help in 2D game programming. For Instance, Graphics2D was really handy to set parameters on the game screen like text and etc.…

Our modelization choices

We decided to set up a main class called **GameGUI** that takes care of configuring all of our game elements, using the functions **run()** which is the game's main loop function, update() and paint() that refresh the positions of the elements and draws them on our map.

The main elements of our game are Player.java that represents the player and ibol.java which is the name we gave our monster. Both classes are subclasses of the class entities.java which contains the default "**blueprints**" for every entity created (**hitbox, image, HP**) with each their movement and draw functions.

For the map, the class **TileMang** defines the different tiles (wall, floor, golden tile, etc) in a 2d array of type tile and gets their images as well as sets their collision attribute to true if necessary.
The map data is stored in a txt file containing numbers each representing a different tile.

Tiles design

(all were designed by us)
Main algorithmes 16x16

Changing map and monsters :

In order to create multiple monsters and put them in different rooms, the class Assets lets us create the monsters by giving them a starting position, and putting them in different arrays of type Entities in the function **setOnMap()**. Each array corresponds to a specific room. Then **setOnMap()** is used in GAmeGUI.java in **setupGame()** method.

To move from one room to another, the algorithm that we came up with detects the player's position on the map and load the next room if the player reaches the coordinates of an "**exit**" tile of the initial room with a switch statement that takes in the player's level.

Whenever a room is loaded, the monsters do not move until the player has taken their first step

collision

In the **checkTile** function, that takes the player entity as a parameter and calculates its hitbox to check for collision between it and other elements of the game such as the walls and the monsters.
This is done by getting the four solid sides of the player character, and depending on his direction, if the next tile is a wall (or a **monster**), the attribute **collisionMode** of the player is set to true and he can no longer move in that direction.