Extensions

I. Animation

We have created a class Animation for this extension. This class's constructor takes in as parameter many things (c.f. Javadoc) but most importantly, an ellipse of Strings spriteNames.

If an instance of this class is initialized with only one spriteName, then the image referenced by this name is cut up into 16 smaller images (and then converted into sprites) (eg. the case of the main player). These 16 sprites are shown turn by turn (when appropriate according to the orientation) to create the animation.

However, if the number of spriteNames during initialization is more than one, the referenced images are not cut up and the animation class just draws these sprites turn by turn.

Any class wishing to have an animation just needs to have an attribute animation and initialize it appropriately. Also, in the draw function of this object, the draw function of its attribute animation should be called with the Canvas and the object's Orientation as the parameters.

II. Following character

We have implemented a new character in the Enigme game – a character which follows the main player wherever he goes. This follower also mimics the movement of the main player – that is, his animation is synchronised with that of his parent (main player) and so is his speed.

III. Foreground

This class was implemented in an analogous way to the class Background. However, the depth of its attribute sprite (ImageGraphics) was changed to 999 so that it is drawn above all other graphics (except for one case described below).

With the help of this class, the player can now go (and hide) behind trees in the Enigme0 map (unless he is flying, explained below). (We also added a foreground to other areas, but they are just empty.)

IV. FastShoes and Running Mode

The class FastShoes (extending Collectable) was implemented with a corresponding sprite. When the EnigmePlayer collects a pair of FastShoes (a private attribute of the EnigmePlayer hasFastShoes becomes true), he can enter/leave the fast motion mode by pressing the key W at any time.

The running mode is implemented by changing the animation duration of the player's movement and adapting the animation delay of the player's animation.

V. Dialogs – (Class Dialog given, Interface Talker implemented)

We have implemented an interface which forces any class which implements it to redefine the method Dialog getDialog() which returns a dialog proper to the class. The interface Talker also helps develop a common link between all the classes which show a dialog.

Some classes which want to show a dialog like Collectable and Switcher implement this interface.

The EnigmePlayer then shows draws these dialogs when he **CAN** interact these objects.

VI. Pause

The class Area has private boolean attribute “paused” which indicates if the game is paused or not. When the user presses the Space Key, the attribute is negated. If the game is paused, the update function of the area does nothing but draw the actors. A text graphics is also shown saying “Game Paused -> Press Space to continue where you left off.” The parent of the text graphics is the view candidate of the area and is anchored to be just below the view candidate.

VII. Sage

We implemented a new class Sage which is a non-moving character in the game who gives information to the main player at different stages. Corresponding sprites were used to draw him. Whenever the player interacts with the sage, the sage automatically turns to face the player to give him the information.

Extension -> We also implemented a class **SignalSage** extending Sage. These signal sages act as Logic Signals – they are “On” when their orientation matches a “Correct Orientation” defined during their initialization.

VIII. Egg and Flying Mode

We have implemented a new class Egg extending Collectable which represents just an egg with an appropriate sprite.

The speciality of this collectable is that once the player collects an egg, he gains the ability to fly. The user can simply press F at any time to enter/leave the flying mode.

When the player enters flying mode, he can move over all types of cell (no restriction except for the fact that he cannot go out of the grid). The sprites of both the player and the follower are adapted to become sprites of birds (which are also animated).

We also had to take care of the fact that when the player is flying, he should be visible above the foreground (that is, the trees and the house in Enigme0). To implement this, we just changed the depthCorrection of the flying sprites to Float.MAX_VALUE so that their sprites are drawn above everything else in the game.

IX. Moving rock

We also added a class MovingRock to the game. These are moveable rocks which move in the direction corresponding to the orientation of the player when interacted with.

Moving rocks are also Interactors because they can interact with pressure plates and pressure switches.

Extension -> We also made an extension of this class called SignalMovingRock which have an attribute Logic. So a signal moving rock is basically a moving rock which can only be interacted with if its signal is on.

X. Bonfire and Signal Bonfire

A class Bonfire was also implemented with its appropriate animation.

Extension -> We also made an extension of this class called SignalBonfire which is basically a bonfire which is on when its signal is on. It also acts as a Logic so that its state can be used for other signal-based processes in the game.

XI. Classes Collectable and Switcher

Classes collectable and switcher were implemented as suggested in the handout. These classes regroup many classes like egg and apple (collectable) and torch and bonfire (switcher) permitting better encapsulation and avoiding repetition of code.

They both implement the interface Talker as all the objects inheriting from these classes want to show dialogs telling the use how to interact with them. (eg. “Press L to turn me on/off”)

Bugs and Possible Improvements

I. The extensions follower and the running mode aren’t perfectly compatible with each-other. There is a bug produced when the current mode (running / not walking) of the player is changed while he is in motion – the follower’s relative position is changed a bit as the follower moves too fast/slow during that frame.
II. When a moving rock is pushed into a cell whose space is taken, the rock doesn’t move into it but, for some unknown reason, it becomes un-interactable and its cell space is not taken anymore (the player can walk into it).
III. For now, we don’t prevent the player from changing from flying mode to walking mode at any time. This means that one can fly onto water (or a wall) and then change to walking mode (but he won’t be able to move anymore unless he changes back to flying mode). But this problem can be fixed without too much trouble.