Kenneth Li

APCS Period 3

Mr. Platek

01/25/2016

Final Project Report

1. Description

This project is a 2D puzzle platformer game utilizing the Graphics2D and Swing libraries. The player must maneuver a character from the beginning of a series of platforms to the green goal platform. Other types of platforms in this version are fire platforms, which kill instantly, and bouncy platforms, which launch the character high into the air.

1. Instructions
   1. Begin by compiling all the classes with “javac \*.java” and running the driver class (which also happens to be the JPanel) with “java Game”.
   2. The console will display messages to the user. To move, use WASD, where W is for jumping, A and D are for lateral movement, and S is for falling quickly.
      1. Up to two jumps are available: one directly off the ground and one in the air – if you are already in the air, you are permitted only one jump.

(Note: The double jump also cancels other vertical velocities, such as those imparted by bouncy platforms.)

* 1. Reach the end of all the levels to win!

1. Design

This project was designed specifically to be extremely flexible and extensible. The brunt of it is a platformer engine, which includes a physics engine, a graphical interface, a game loop, and a map creator.

The physics engine, which manifests itself in the various update methods (especially Entity and Player), uses six variables to represent two time-derivatives of displacement in two dimensions: \_vx and \_vy describe the velocity; \_ix and \_iy describe the input by the player, which factors into final rate of change of displacement but does not directly change the velocity; and \_ax and \_ay describe the acceleration, or rate of change of velocity. These variables all come together to change \_sx and \_sy, which directly determine the coordinates of the character.

The graphical interface includes components from the Swing and Graphics2D libraries. A JFrame, stored in the \_frame field of the Game class, holds the Game, itself an extension of the Swing JPanel. The Game's painting methods call render(Graphics) on the platforms, stored in ArrayList \_platforms, and the player, stored in \_player. These objects' render(Graphics) methods, as outlined in the abstract class Tangible, render themselves with the fields \_sx, \_sy, \_width, and \_height. Their colors are determined by the individual subclasses.

The game loop uses a while loop controlled by a boolean \_isRunning to allow periodic updates at a fixed rate of TARGET\_FPS (default 240) frames per second. This is accomplished by using calls to System.nanoTime() to space frames of length FRAME\_LENGTH and use that time dt as a multiplier for all physics before rendering. To conserve CPU time, the loop does not update or render if the time elapsed is less than the desired frame length and instead calls Thread.sleep().

The platform maps are created beforehand and stored in plaintext files levelX.txt. The template for levels is described in levelTemplate.txt. These files are interpreted by a Scanner object in loadLevel(int), which creates the player object and the platforms with the data. New levels can be created using the template as a guide.

As said, this game is highly extensible – new levels can be created with the map template and new features can be added with the versatile methods. Future avenues for expansion include more levels, more physics (incorporation of horizontal accelerations), and more platform types (fake platforms that break, moving platforms), as well as more user-friendly graphics (sprites) and a better user interface (dialog boxes instead of console messages).