**DEVELOPERS GUIDE**

Faraz O, Faraz H, Harshan M.

**Overview:**

This document will explain the planning, logic and code that went behind programming the Java game: Lil’ Alchemy. This developer’s guide will suit a developer who might be curious as to what kind of coding went behind programming such a brilliant and complex game as Lil’ Alchemy. The programmer can then get few ideas from our breakdown of few of our classes, and may try to make his or her own game by being inspired by ours. Our game: Lil’ Alchemy is made up of many parts. It has different levels with different maps. A few things we wish we could have made if only we had more time would've been an inventory where the user can create things for his or her character to use and trade with the Sensei. The sensei would have been the wise old man who gives health stat points.

**Features of the Program:**

* Health Bar:

How the Health Bar Works: *Class called Statpoints class contains all the stats of the player and creates an image for the health bar*

Methods: *Game calls a method from the StatPoints class it returns an image that is to be drawn. statpoints ();*

* Enemies (AI Logic):

How the enemies works: *The enemies are controlled by a timer and they start by just randomly facing a different direction.*

How the enemies chase the user: *When the player comes into range of the enemy then the timer for randomly turning stops and another timer is started that makes the enemy chase and attack the player. chase();*

How the enemy looks: *The enemy is an object that extends the Player class so it inherits all the fields from player and methods. It has its own methods specific to the enemy because it is controlled by the computer and not by the user.*

* User (Ash Character):

How the user runs: *The user controls the character by using the left, down, up, right keys and attacks by pressing ‘z’. The player should be able to talk to friends as well and that is done by pressing ‘z’ as well. run();*

How to show the user is running: *The animation for the player is controlled by what arrow is pressed and the animation and movement is done in a timer. The timer allows for smooth, nice animations. The player itself is an object that contains a StatPoints object to hold its stats and also includes pictures for each animation needed.*

How the user looks: *The pictures needed are as follows: Standing position for each direction, a left step and a right step picture for each direction and a punch picture for each direction.*

* Different maps:

How to create the maps: *Game map class contains all the maps and inside of the game map class there is an inner class that is called a ConnectingMap and this holds data field of a Map from mapMaker package and an array of MapConnector from mapMaker package and a String that contains the file name of where the enemies (or friends) are to be loaded from.*

How to create the doors that are linked with them: *The MapConnetor object contains two Point objects and a Map object and every time the player steps on a block, the GameMap checks to see if the Player stepped on a MapConnector and if the Player did then it changes the map based on the Map in the MapConnector the player stepped on and sets the position of the Player to where the Point object in the MapConnector is set to.*

**Suggestions for improvements:**

There could be a few improvements put on our game to make it even better than it already is. In terms of the look of the game, a few improvements can be made. One suggestion could be that we could have made better graphics using picture editing software in order to put more detail into the characters and items. Another suggestion could be that we could have made the whole map using 3D images so that it looks more realistic and we get a sense of depth. Our game would have looked much better and more appealing to potential user/players. Next, our AI logic could have been tweaked, if we were given more time. The enemy interaction could have been improved, by allowing the enemies to shoot at the user, by creating a cleverer and complicated AI logic behind the enemy. Finally, we could have added more features to our game and storyboard. Perhaps we could have added a trade market, where the user would’ve been able to trade the items for other useful items that would help him in the game. Allowing the user to have more options would have surely increased the enjoyment factor in our game. Also, Instead of having Enemy extend Player we could have had an abstract class that is common to both player and enemy classes so then Player would extend the abstract class and Enemy would extend the abstract class not Player class. Also a better way of animating the Player could have been done because now our timer prevents the user from moving when they press the arrow key unless they press it when the timer is not waiting.