SLIME DUNGEON

<< DESIGN DOCUMENT >>

by Yunji Lee

<< PREMISE >>

Player Objective

You are an alien who was traveling across the galaxy when your spaceship crashed on a distant planet infested with slimes. You're currently trapped in a dungeon room guarded by slimes outside. In order to escape, you must collect 4 keys, and the only way to do obtain keys is by feeding stars to the cactus plant in the dungeon room. Stars can be found in the backyard, but you'll need to avoid the slimes while you're obtaining them as they can attack you. Every time the cactus plant becomes fully grown, you will obtain a key. If you can obtain all 4 without dying, you'll be able to escape and win the game.

Game Type: Single-Player

<< GAME ENTITIES >>

Note: Only important entity variables are listed below. Refer to code for more specifics.

Models

1. Entity (extends ImageView) : represents an entity in the game
   int height/width: dimensions of entity's ImageView
   int maxHealth/health: entity's health
   int paneWidth/paneHeight: pane dimensions
   Facing facing: current direction faced
   & Images for facing left/right/up/down

2. Player (extends Entity) : represents player
   int numKeys: # of keys obtained
   int numStars: # of stars obtained
   & Images to control player walking frame animation
   & instances of Cactus, StarsLabel, Keys, HealthBar
   *contains values for facing up/down/left/right

3. Minion (extends Entity) : represents individual slimes
   *only contains values for facing left/right

4. Cactus : represents cactus plant
   int growthLevel: current level of cactus growth (1 - 3)
   & Images for each growth level

5. StarsLabel (extends Label) : represents stars in backyard
   & ImageView to show a single star

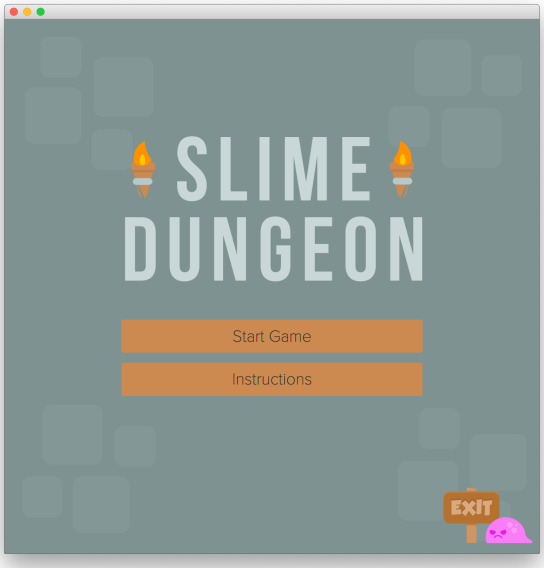
6. HealthBar (extends HBox) : represents health bar
   int maxHealth: player's maximum health value
   int numHearts: corresponding number of hearts
   ArrayList<ImageView> hearts: represents health bar
   & Images for full/half/empty hearts

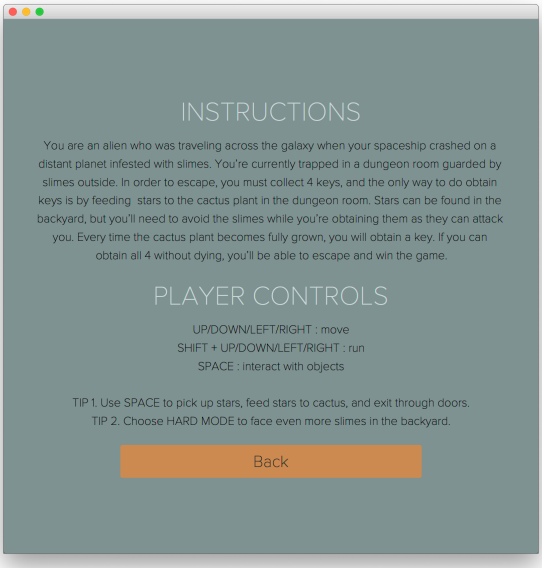
7. Keys (extends HBox) : represents keys bar
   & ImageViews for red/blue/yellow/green keys & placeholders

Views

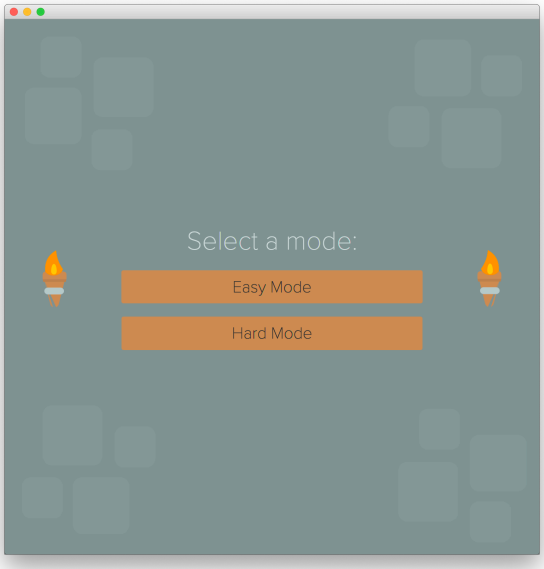
8. BackyardView/Pane (extends GamePaneBase) : represents backyard
   ImageView roomExit: exit to room
   ArrayList<Minion> minions: minions in backyard
   ArrayList<ImageView> stars: stars in backyard
   boolean collision: whether a collision has occurred or not
9. RoomView/Pane (extends GamePaneBase) : represents dungeon room
   ImageView backyardExit: exit to backyard
   Cactus cactus: cactus in room (same as Player's)

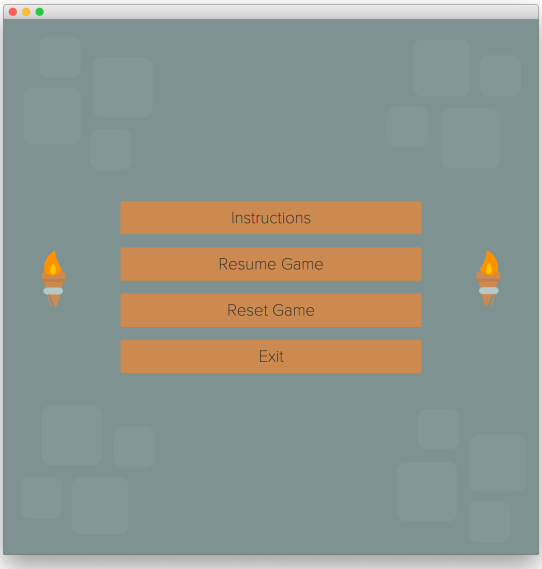
10. StartView : starting/main menu
    Buttons to start game & view instructions
11. InstructionsView : view with instructions & option to return to menu
    Label with instructions & Button to return to menu



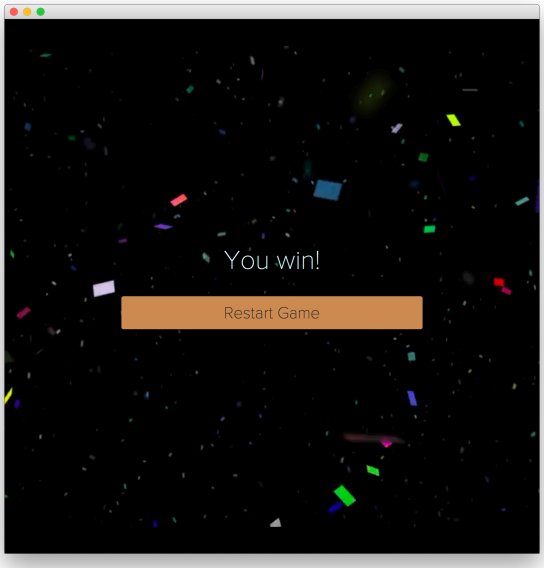


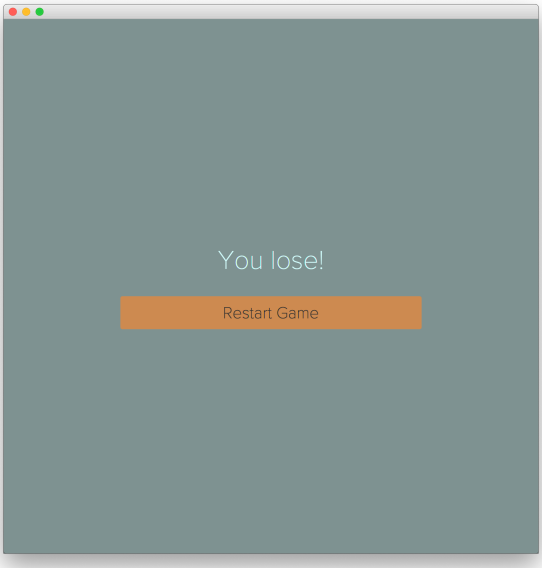
12. ModeView : view with options to play easy or hard mode
    Buttons to start game with easy or hard mode
13. MenuView : in-game view of menu (reset to return to StartView)
    Buttons to resume/reset game, view instructions, exit game





14. EndView : view with either win or lose scene
    Button to return to StartView





Controllers

15. MainController : controls switching between all scenes
    *holds all Views & GameController & MediaPlayer for BGM
16. GameController : controls game play
    GameTimer timer: contains main game loop (AnimationTimer)
    boolean hardMode: controls easy vs. hard mode
    int viewNum: keeps track of which Scene is being displayed
    *holds Player & its associated objects
    *contains booleans to control player movement
    *contains RoomPane & BackyardPane (set by MainController)

<< DESIGN PATTERNS >>

MVC Pattern

As demonstrated in the previous game entities section, the classes are split into packages representing the model, view, and controllers of the game. The model holds all data, states, and game logic. The view represents a presentation of the model. The controller takes user input and interprets its meaning in terms of affecting the model.

Singleton Pattern

This pattern is utilized for the GameController class, as there should only ever be a single instance of it. A global point of access is provided in such a way that every access to the instance returns the same instance.

Template Pattern

This pattern is utilized in the Entity abstract base class. The class contains a reset() template method set to be final so it cannot be overwritten. The reset() method contains an abstract method called resetEntity() that does nothing in the Entity class, but must be implemented in the subclasses (Player/Minion) as each contains different private variables that must be reset. Thus, reset() defines the skeleton of the reset algorithm, deferring some steps to subclasses.

<< GAME PLAY >>

User Controls

UP/DOWN/LEFT/RIGHT : move player
SHIFT + UP/DOWN/LEFT/RIGHT : make player run
SPACE : interact with interactable objects

Interactable Objects

Note: User can interact with objects by moving on top of them (i.e. intersecting).

Room

Cactus: to feed it stars
Backyard Exit: to exit to backyard

Backyard

Stars: to pick them up
Room Exit: to exit to dungeon room

Modes

Easy: number of slimes starts at 10
Hard: number of slimes starts at 15

Rules

- Player starts with a total of 8 health (4 hearts).
- Player can move back and forth from the dungeon room to the backyard through the respective doors.
- Every time the player enters the backyard, the minions and stars are reset (i.e. the full set of stars reappear in random locations).
- Slimes move randomly (horizontally) across the backyard and attack the player upon collision.
  - Player loses 1 health (1/2 heart) per collision.
- Player can feed stars to the cactus.
  - Cactus becomes fully grown after 3 stars, and player will obtain a key. Cactus returns to original growth stage.
- For each key collected, the number of slimes in the backyard will increase.

End States

Win

- Player wins by collecting all 4 keys without dying.
- Will be led to EndView with a win message and an animated confetti background video.

Lose

- Player loses if their health reaches 0.
- Will be led to EndView with a lose message.

<< REQUIREMENTS >>

Requirement Options Implemented

1. Add sound to your game using the JavaFX Media classes.
   Background music added in MainController.
2. Add video to your game using the JavaFX Media classes.
   Confetti video added to background of winning EndScene.
3. Animate 2 “non-player characters” (NPCs) with different types/ movements for those entities.
   Multiple slime NPCs move around the backyard randomly & independently, attacking player upon collision.
4. Create at least 2 different levels (easy level & hard level).
   Easy and hard mode with varying numbers of slimes & considerable difference in difficulty.
5. Make the world bigger than the window size.
   World display updates when player character exits through a door.

Special Topic: JavaFX Frame-Based Animation

I used frame-based animation to make my sprite (Player) appear to be walking when moved around the screen by the user. For each direction it can face, it has a sequence of 3 images, stored in the Player class, that when swapped in rapid succession by the AnimationTimer in GameTimer, creates the illusion of the sprite walking.