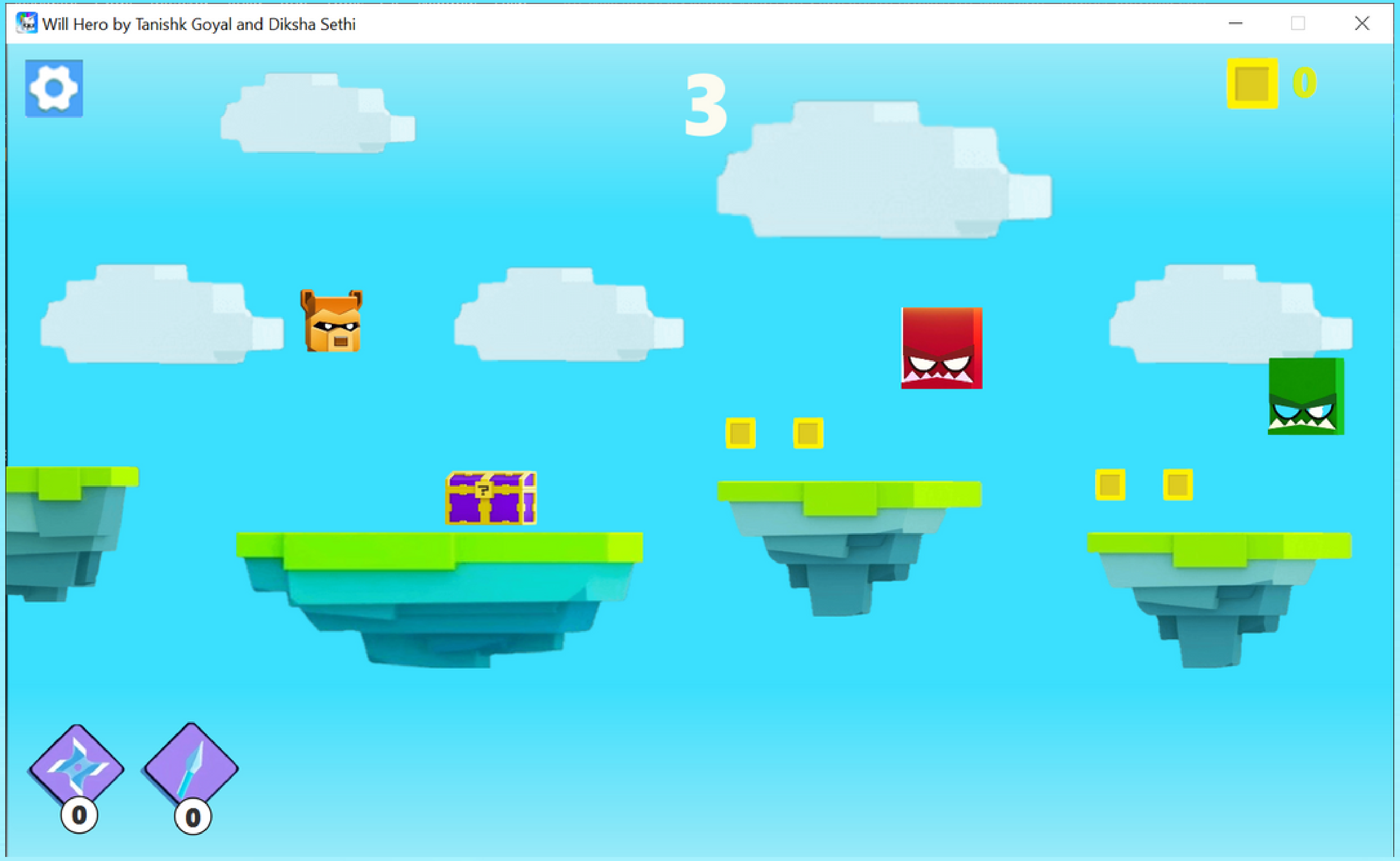
WILL HERO

By - Tanishk Goyal and Diksha Sethi



Design Features and Implementation

- Solid Class - gives the properties of "matter" to any object
- Collidable interface marks the object to have trackable collisions - Hero and Orcs
- We use rectangles to define the collision area of platforms
- Implemented physics in the game using gravity and kinematic equations
- Using Physics, the game has the ability to track any collisions between all Solids
- Used JavaFX timelines for:
- Game Physics (Perpetual Jumping and Collisions)
  Moving Hero Forward on Click
  Using Weapons
- Obstacle - Falling Platform
- Weapons - Throwing Knives and Shurikem

Challenges Faced

- Challenges : Handling too many images caused laggy animations, Solution : Created Staging Area, Garbage Area and Collision Area demarkations while iterating through objects, only staged objects that were in frame
- Challenge: Keeping track of individual collisions was too chaotic
- Solution: Created hasCollided function in Solids and collidesWith function in Collidable, hasCollides checks if collision with passed Solid has taken place, if yes, it returns
- 1 - Collision with Left
- 2 - Collision with Top
- 3 - Collision with Right
- 4 - Collision with Bottom
- This information is then passed on to collidesWith function, which implements the consequences of collision with that side

Design Patterns Used

- Observer : The EventHandler waits for a keyboard input and updates the game objects' state accordingly
- Singleton: Created single instances of Helmet and the Weapon slots, which can only be accessed through getInstance()
- Factory: Weapon Slots create the weapon objects assigned to them, the game class only calls the useWeapon() function of these slots

Individual Efforts

Tanishk Goyal (2020141)

- Created Weapons and Framework for shooting
- Designed and made the collidesWith Mechanism
- Made working implementation of Chests
- Designed and Coded the physics of the game
- Completed Resurrect Hero
- Set Up JavaFX elements in Panes
- Designed the stagingArea, garbage Area, removeDeadThings frameworks
- Debugged Diksha's Code

Diksha Sethi (2020056)

- Created the framework of the classes according to the UML diagram
- Designed the Collision conditions
- Generation of game objects and obstacles randomly
- Implementing the Falling Platform
- Serialization and Deserialization
- Working of the restart, load game, save game, play game
- Collected and designed all the visual components of the game
- Debugged Tanishk's code :)

Bonus Features

- Implemented Physics in hero movement with gravity and equations of motion
- Given 2 ways to play the game, keyboard (Space Bar) and mouse
- Orcs, Chests and Coins are randomly generated
- Platforms can be updated in Scenebuilder to customize level, and the code still works
- Hero can jump on top of orcs, and all other possible collisions are implemented
- Information about loaded games, can be viewed before loading the game
- Smooth moving animations for all hero jumps and forward movements
- Ability to switch between weapons using number keys 1 and 2
- Orcs can push each other back on collision, to give real life animation feel
- Created Load Game, Save Game, Resurrect Hero Options using pop up menus