CS121: The Saucer Game Project

In the saucer game, the player moves a flying saucer in one of four directions in an attempt to land it between two buildings. The saucer crashes if it touches the ground anywhere outside of the two buildings or if it touches the top or either side of the frame. The also saucer crashes when it touches either building.

The saucer starts in the "hovering" state, and ends in either the "landed" or "crashed" state.

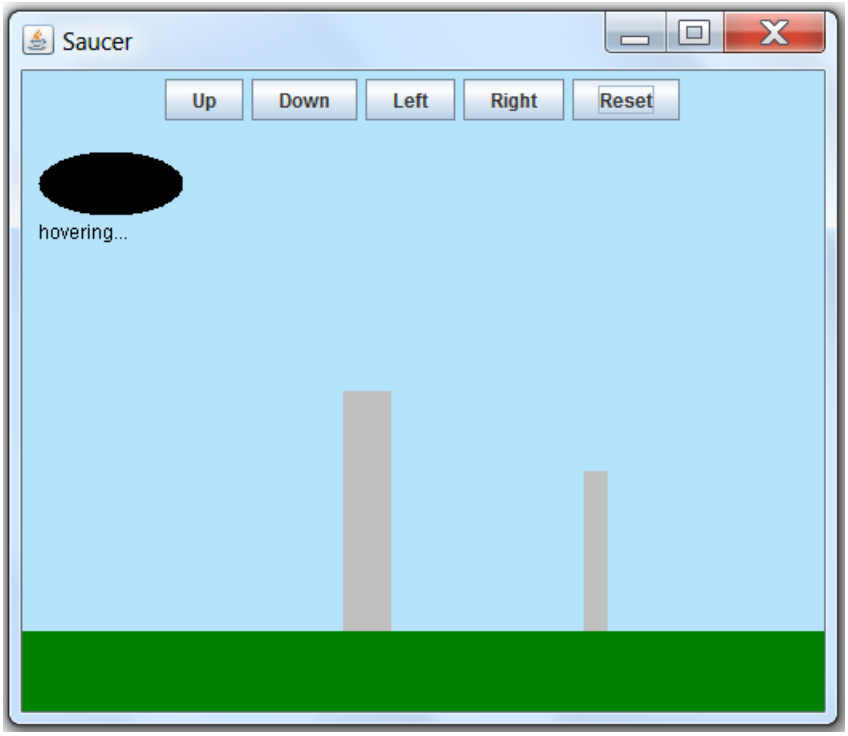

Figure 1: Starting view

The saucer is moved by the player clicking one of the directional buttons: up, down, left, right. The saucer moves by itself in the direction selected until another direction is selected. The saucer only stops moving if it lands successfully or crashes. The game can then be reset and play begins again.

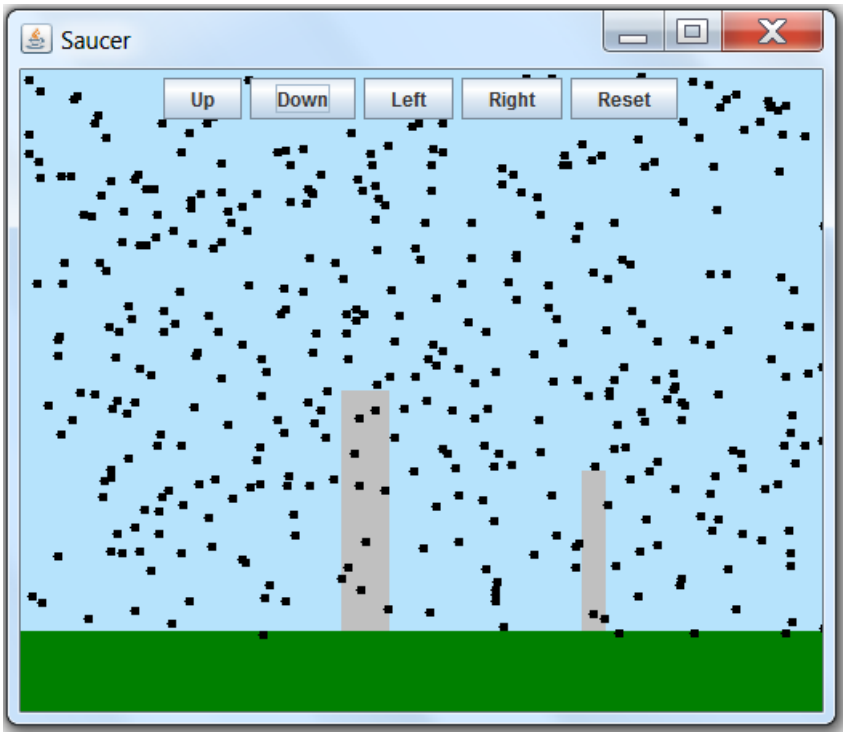

Figure 2: Saucer crashed

The code:

Take time to look over and understand the code before you start your own coding. The important dimensions, colors and other values are defined at the top of the class for easier use below. If any values have to be modified, they can be changed in one place. All position and boundary calculations are easier to understand due to the names of the variables. The state of the saucer is defined by a boolean variable "hovering", which is true at the beginning of the game. If the saucer encounters the ground outside of the landing area, the other sides of the panel, or the buildings its hovering status becomes false. The String variable saucerStatus is used to display the saucer's status below the saucer. This variable can have two values: hovering or landed. The direction of the saucer is also stored in the direction variable, a String: "up", "down", "right", "left".

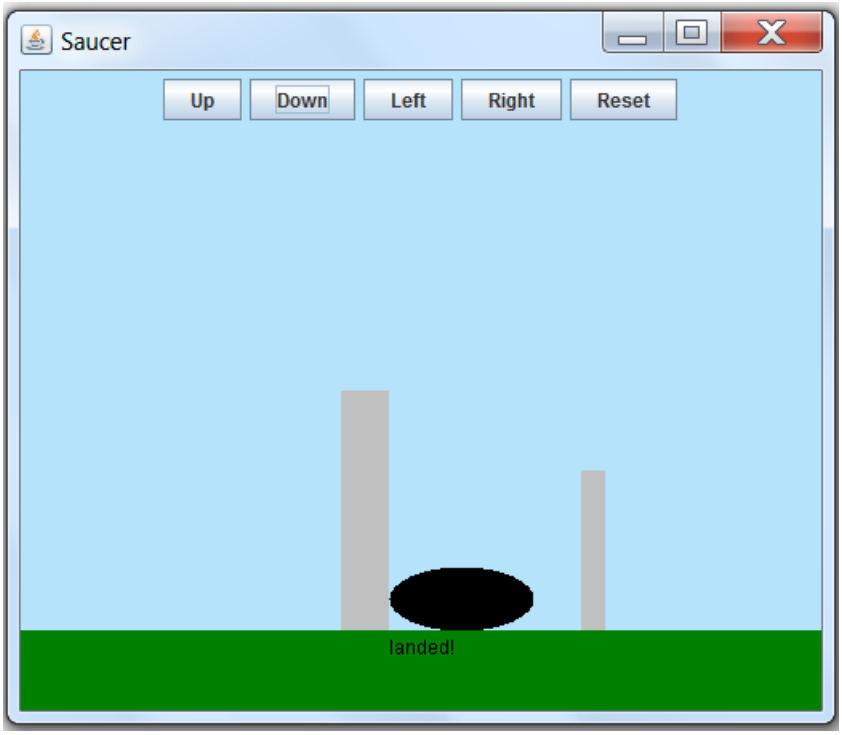

Figure 3: Saucer landed successfully

Positioning and boundaries of the saucer are determined by the rectangle that surrounds the oval saucer shape. The Java Graphics object actually draws the oval within a rectangle:

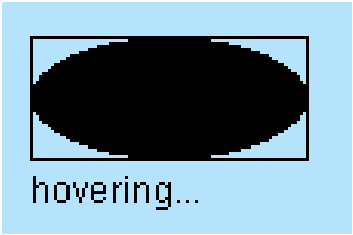

Figure 4: Saucer boundaries

The position of the saucer is defined by the upper left corner of the rectangle. When you want to calculate the saucer's position you add either the saucerWidth or saucerHeight to the saucerXpos and saucerYpos depending on which direction and boundary you are interested in.

The Timer object is used to move the saucer. At the start of the game, the timer is off until the player clicks a direction button. The code picks up on this event and sets the direction variable appropriately and starts the timer if it is not already running. When the timer is running, it produces events every 100 milliseconds. These events are handled in actionPerformed. First, the direction needs to be determined (this was set when a direction button was clicked), then the move is made (using the saucerMoveIncrement value, defined at the top of the class).

Finally, the new position has to be checked and the status of the saucer is adjusted accordingly. Make use of the saucerTouchingBuilding method provided when appropriate.

Note that in the paintComponent method, the hovering status is checked, so if the saucer has crashed it will be picked up there for the proper rendering (a crash!). The timer is stopped when either a crash or successful landing occurs.

The rest button stops the timer and sets all program values to the beginning state.