CS1331 Homework 8

Point-and-Click Adventure

Due Saturday November 3rd, 8PM.

This homework will be more free-form than the others. You are encouraged to plan out your code structure before you begin, so you can see how all the parts come together.

The Big Picture

In this homework you will be making a simple point-and-click adventure. A point-and-click adventure is a type of game where a user is shown a scene (as seen above). They can click on doors and objects with the mouse, and then something happens. Usually, the player has an inventory, and some sort of narrative is being told during the adventure. We won't require you to make a fun adventure game, but we do encourage you to get creative.

Please read the whole guide thoroughly before you begin. If you have any questions, please ask them first on Piazza. Check Piazza often, as we offer clarifications and advice there.

Functionality Requirements

Here are the general requirements. They will be elaborated on in the next section.

- At least 3 persistent scenes (locations)
- A way to travel between scenes (doors, etc)

    - You must be able to eventually reach every scene.
- The player must have an inventory.
- Objects should be strewn around the scene.
    - At least one object per scene.
    - The objects must be collectable.
- The game must have a win condition (you may make the win condition collecting all of the objects, if you like).

Specific Design Requirements

Scenes/Locations

The adventure should have at least 3 scenes in it. These should be unique locations that look different from each other. You may draw them yourself, or use photos you find on the internet (make sure to list your sources in your submission). The scenes should have doors (some sort of portal) that, when clicked, allows the user to go to the next scene. The game will begin with one of the scenes that you have set up to be the “starting scene”. In each scene, there will be objects that the user can click on. Upon clicking on the object, it will get added to the user's inventory. The scenes must persist even if the user leaves them and comes back. This means that if something changes in a scene—example, an object is picked up by the player—it will have to remain that way if the player leaves and comes back.

Player Inventory/ Objects in the Scene

The player will have an inventory, that he can fill up with objects they can pick up from the environment. When the user clicks on the object, it will be removed the scene, and be placed in the user's inventory. You may make something happen when an object is picked up (like displaying a message, etc). The player can't interact with their inventory, so don't worry about having to program a clickable inventory. At the top left of the screen, though, there should be some strings/images drawn on-screen that show what is in the player's inventory.

Win Condition

The game should have some kind of win condition. What it will be is completely up to you, barring that one condition is met: it must involve visiting every room and interacting with some objects.

Bringing it all Together

You are free to use any design you like (we will grade it according to the style we've taught you in class). You may hard-code things like the number-of-scenes, and other essential parts of the game logic. You may also have a class with static methods and variables that can control the game flow (store current-scene, player inventory objects, etc).

Sample Code Structure

In case you need some help getting started, here's a simple code structure you *could* use:

- GameController.java
  - This is the controller of your application.
  - It has a list of scenes, in a set order.
  - It has a reference to the current scene.
  - It has a static method for switching the current scene.
  - It has a reference to the GamePanel, in case it needs to let the panel know to repaint() itself.
- GamePanel.java
  - A panel with a MouseListener attached. When a click occurs, it lets the current scene (found in GameController) know where it occurred. Remember, you need to call *repaint()* on the panel when you need it to redraw itself. It should also draw what is in the player's inventory, somewhere on screen.
- Door.java
  - This object takes in a Rectangle, and an integer that represents which scene to jump to when it's clicked on. It should have a method that takes in an x and y coordinate, and decides if that point is within its Rectangle. If so, tell the GameController to switch scenes. We will choose to put an image of the door in the background image of a Scene, so there is no need to have a Door store an image.
- Pickup.java
  - A pickup is just like a Door, only it will have an image, and it will not usually switch the scene when it is clicked. When a player clicks on a Pickup, it should display a message for the user, and then be removed from the Scene, and placed in the player's inventory.
- Scene.java
  - A scene contains a background image, a list of Pickups, and a List of Doors. When the GamePanel detects a click, it should let the scene know, and the scene should let every Door and Pickup know where the click took place, and let them all react accordingly.
- Adventure.java
  - Set up all the Pickups and Doors, and use them to build Scenes. Build the GUI, setup the controller, and start the game.
  - Every phase of building the adventure will happen in a separate method, all called, in order, from the main method. Initially, all of our components (Scene, Pickup, etc) are just blank slates, but we give them the values and images they need, and assemble them properly into meaningful scenes.

The README

Your submission should also contain a README, describing how your TA can finish your adventure. It should be detailed enough so that they can follow it and complete your adventure quickly. You should also put your image sources here.

Turn-in Procedure

Turn in the following files on T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft. All .java files should have a descriptive javadoc comment.

Don't forget your collaboration statement. You should include a statement with every homework you submit, even if you worked alone.

Verify the Success of Your HW Turn-In

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**
    (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework (past the grace period of 2 am) will not be accepted regardless of excuse. Treat the due date with respect. The real due date and time is 8 pm Saturday. Do not wait until the last minute!

Extra Credit

You may only get a total of +10 extra credit points.

- (+5) Clickable Inventory: The user can click in their inventory and the object they click on will respond with a message describing it.

- (+5) Save/Load: The ability to save/load your adventure at any time. Inventory and pickups in a scene should also be preserved.
- (+5) Puzzles: Your adventure can have a clever puzzle that involves the altering of a scene somehow. Example: clicking on a cabinet somewhere in a scene will cause the cabinet to open, revealing a key that unlocks a door. Be sure to mention your puzzle explicitly as an extra credit puzzle in your README.