Tetris

System Requirements Document created by Jared Giroux.

Document is based on the PeRFUMe software quality model, developed by Jared Giroux.

This document was created for personal use as a planning resource for a school project. It is not to be distributed or disseminated for professional use or profit.

* Requirements

Portability

* + - Installability
      * The system must be fully functional within its development environment
      * The system should be fully functional within other operating systems environments, provided Java libraries from 2020 are installed
    - Adaptability
      * The classes used by the system should be able to be used in other environments with no reliance on other classes from this system.
    - Compatibility
      * The system shall not interfere with the performance of other systems in its environment.
    - Availability
      * The system should be transferable through 1 jar file to other environments.
    - Replaceability
      * The system shall not leave artifacts after uninstallation.
  + Reliability
    - Fault Tolerance
      * The system should not crash upon interaction with any part of the screen.
      * The system must not crash if an unsupported file (such as a txt file) is chosen by the user (such as when a save file is chosen).
    - Recoverability
      * The system should update the “runtime” file (i.e. the file edited during the time the system is running) containing its profile after each edit made by the user.
      * The system should be able to save profiles to different files stored in the environment, reducing the impact of a system or environment failure.
    - Integrity
      * The system should not be editable by users.
    - Confidentiality
      * The system must not store any data other than saved colors.
      * The system must have read access to pictures stored in the environment, without any ability to store data from those pictures other than a color that is saved.
    - Authenticity
      * N/A
    - Accountability
      * N/A
  + Functionality
    - Accuracy
      * The system must behave in the same manner as other tetris installations do
    - Appropriateness
      * The system’s functions will be to run tetris and record high scores.
    - Compliance
      * The system should adhere to the coding conventions listed here: <https://kotlinlang.org/docs/reference/coding-conventions.html>.
    - Time Behavior
      * The system should respond to user input within 0.1 seconds in all situations.
      * The system’s launch time (i.e. the time it takes to start running) should be under 1 second.
    - Resource Utilization
      * The system should not require more than 1 gigabyte of storage from its environment.
      * The system should not require more than 2 gigabytes of ram from its environment.
  + Usability
    - Learnability
      * The system should have a tutorial screen that displays the controls for the game.
    - Ease of Use
      * The system should have a depth of 3 commands. Depth refers to how many choices the user can make going from the main menu before reaching a screen with no choices other than to return.
    - Attractiveness
      * The menus of the system should be composed of rectangles with rounded edges.
      * The system’s color scheme should be composed of light pastel colors.
      * The system’s font should be sans serif and not stylized (i.e. not cursive or textured).
    - Recognizability
      * The main functions of the system shall be visible and available from the main menu shown when the system is launched.
    - User Documentation
      * Refer to Learnability.
    - Technical Accessibility
      * The system may have colors that are friendly to users with color blindness
  + Maintainability
    - Development Documentation
      * The system must have tests for each function and feature of the system, with an explanatory comment for each test.
      * The tests for the system should be documented using json files.
    - Testability
      * The system must not have a feature created for it without first having a test written for that feature.
    - Modularity
      * The system should be created using object oriented programming, i.e., there should be low to no level of coupling between the application’s functions.
    - Modifiability
      * The system must have tests that ensure the overall functionality of the system after each change made to the system.
    - Analyzability
      * The system should have tests that check the performance of the system, to see if it meets the criteria listed under time behavior and resource utilization within this document.