Use Cases

Use Case Number:

1

Application:

Graphic 2D Modeler

Use Case Name:

User can modify or remove shapes

Use Case Description:

User could add, remove, or change shapes to the 2D modeler based on their status as either a regular user (No permissions), Pro-user, or Administrator. The following shapes are could be modified/ removed in the modeler: Line, Polyline, Polygon, Rectangle, Square, Text, Ellipse, and Circle. The user could define information/ position of the shapes as well as specifying the color and line width of said shapes.

Primary Actor:

The user (Regular, Pro, Administrator) and the 2D modeler system.

Precondition:

Regular user (No permissions): No Prior conditions must be met.
Pro User: User must log in with a username and password with matching permissions
Administrator: User must log in with a username and password with matching permissions

Trigger:

User selects one of predefined shapes, fills in orientation, position, color, line-width, and dimensions to specific to shape. Once finished, user selects the “Update” button and the shapes are than added to the 2D modeler.

Basic Flow:

With valid information and login permissions (Pro user or Administrator).

1. User triggers event with “Update” button.
2. Constructor of specified shape is called with the information added by user.
3. Shape is stored in shape vector.
4. Menu refreshes and new shape is displayed.

Alternate Flow:

User does not posses permissions of Pro user/ Administrator or could not login.

1. User is not shown the information required to be entered for a shape object and thus cannot update the program.

User enters the incorrect information for either the dimensions, color, line width, or position/ orientation of a specific shape object

1. User presses “Update”.
2. Shape object will neither be created nor modified.
3. System outputs error message to the use.

Use Case Number:

2

Application:

Graphic 2D Modeler

Use Case Name:

2D Modeler System startup/ boot

Use Case Description:

User clicks on 2D-Modeler icon to start the program. The saved information of the previous state of the program is saved/read from an internal .txt file.

Primary Actor:

The user (Regular, Pro, Administrator) and the 2D modeler system.

Precondition:

No preconditions.

Trigger:

User selects the program icon and the program than builds.

Basic Flow:

.txt file values/ info is valid and file is not corrupted

1. User triggers the startup by selecting the program's icon.
2. .txt file is read into program via system .txt file parser.
3. Shapes are displayed based on information parsed from .txt file.

Alternate Flow:

.txt file values are incorrect or file is corrupted

1. User triggers the startup by selecting the program's icon.
2. .txt file is read into program via system .txt file parser.
3. No shape is created/ added to the vector.
4. Error message is displayed for the user.

Use Case Number:

3

Application:

Graphic 2D Modeler

Use Case Name:

2D Modeler System shutdown/ close

Use Case Description:

User selects the “Save and Shutdown” button and the information is subsequently saved to the .txt prior to system shutdown.

Primary Actor:

The user (Regular, Pro, Administrator) and the 2D modeler system

Preconditions:

No preconditions

Trigger:

User selects the “Save and Shutdown” button in the programs primary window.

Basic Flow:

1. User selects the “Save and Shutdown” button to trigger the save function.
2. Program then parses the current state of the shapes vector and the information it contains overwriting the .txt file to save current progress.
3. Program deallocates memory and shuts down.

Use Case Number:

4

Application:

Graphic 2D Modeler

Use Case Name:

2D Modeler System feedback

Use Case Description:

User selects the feedback tab and enters in feedback regarding the software and its performance.

Primary Actor:

The user (Regular, Pro, Administrator) and the 2D modeler system

Preconditions:

No preconditions

Trigger:

User selects the feedback tab on the top of the window.

Basic Flow:

1. User selects the tab at the top of the window as a trigger.
2. User types in feedback on program
3. User then selects save
4. Feedback is saved to internal memory for developers to review.