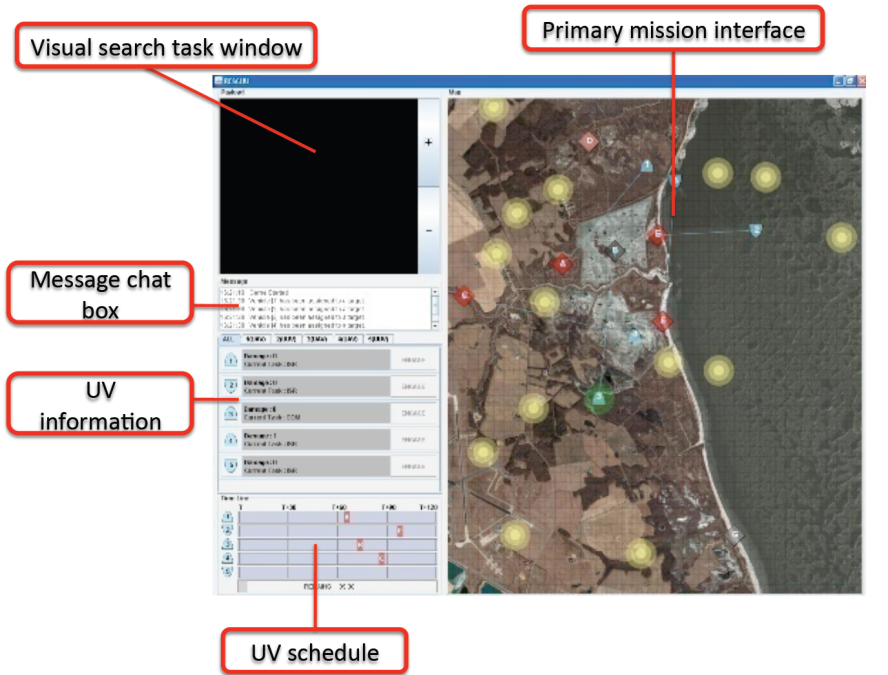
Guide to RESCHU

Overview of RESCHU

The RESCHU simulation allows an operator to control a team of UAVs. UAVs are directed to targets, and upon reaching a target, the operator is presented with an image and is prompted to look for a specific object within the image. Three types of decision support systems place constraints on the user:

* No retrial: the operator must select a location on the image to continue to another target
* Retrial with consent: the operator is prompted to move on to a different target after a given period of time
* Retrial without consent: the operator is forced to move on after a given period of time

For each decision support scenario, the operator can also be given a timer that displays the recommended amount of time remaining for the search task.

The graphical user interface displays all mission information to the operator.

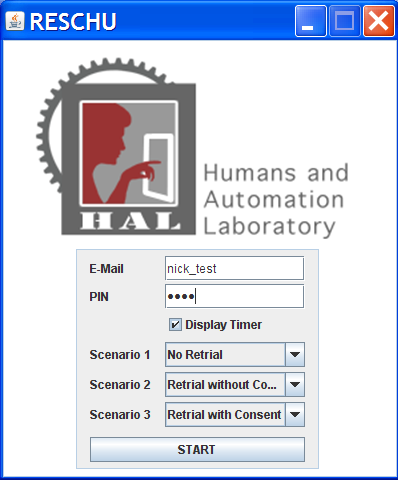
In recent versions of RESCHU, a path planning algorithm has been used to assign vehicles to targets. The implementation of this algorithm is ongoing and will result in modifications to the primary mission interface.

Overview of Classes gives a general description of the features of the most important classes in RESCHU. See API documentation for a more detailed description of specific methods and attributes.

Overview of Classes

**reschu.app.AppMain:**

**The method AppMain.main is called to begin the execution of RESCHU. AppMain.main ensures that a connection can be formed with the database, and if so, initiates a connection with the database and creates the login frame.**



**When the user enters his/her data into the login frame and clicks the start button, the username and password are used to login to an account on the database. A Reschu object is instantiated, and takes control of the follow of the program.**

**reschu.game.controller.Reschu:**

Reschu is the JFrame that displays the GUI, and is as well an important class in containing the other objects that are Components in the GUI or are important aspects of the game engine. The Reschu object serves to connect the other faucets of the simulation, and performs writing to the database.

The Reschu constructor calls the initComponents method, which instantiates many of the objects contained in a Reschu object, including the Game object and screen Components. Control of the program passes to the Game class.

**reschu.game.model.Game:**

**The Game class serves as the engine for the simulation, causing events to unfold in time. The Game class includes a Timer object that generates an ActionEvent every 500 milliseconds, which is used to keep track of time in the simulation and initiate actions that occur as a function of time, such as target arrivals, prompting in the visual search task, and keeping track of time remaining in the simulation.**

**The AutoTargetAssignAll method is used to give vehicles their target assignments.**

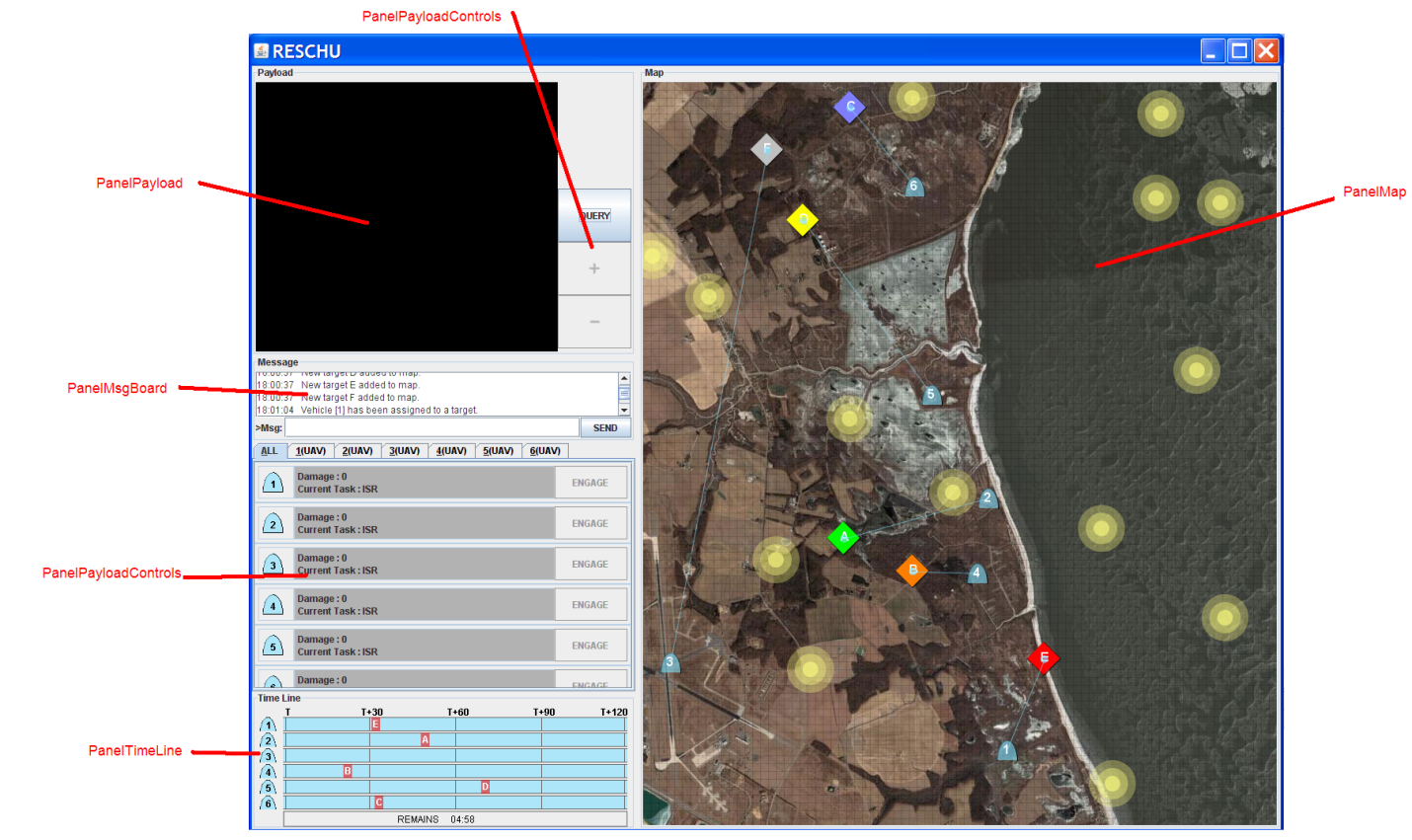
**reschu.game.model.Map:**

The Map class deals with aspects of the simulation that are dependent upon their locations in the environment. A Map object contains all targets, hazards, and vehicles, and is responsible for generating and removing targets and hazards from the environment. The Map class differentiates between assigned, unassigned, and temporarily unavailable targets.

**reschu.game.utils.Optimizer:**

The Optimizer class handles all optimization features used in RESCHU. Optimizer is used for solving the travelling salesman problem to generate an optimized assignment of UAVs to targets. The optimizeAssignments\_alg3 method implements a receding horizon assignment, and optimizeAssignments\_alg4 implements a receding horizon assignment of each vehicle over its assigned cluster(s). The Optimizer class also contains methods for performing clustering of LocationObjective objects and for performing Monte Carlo simulations.

**Elements of GUI:**

****

**reschu.game.view.PanelControl:**

Displays information specific to the vehicles and each one’s status. Can be used to engage a vehicle at a target.

**reschu.game.view.PanelMap:**

Displays locations of targets and vehicles, as well as vehicle paths. Serves as a listener for mouse interactions with this part of the GUI. From the PanelMap, the user can engage vehicles, as well as change vehicle assignments to targets and generate waypoints.

**reschu.game.view.PanelMsgBoard:**

Is used to display messages to the operator.

**reschu.game.view.PanelPayload:**

Displays the visual search task. The user can pan the image by left clicking and can select a location by right clicking. When a location is submitted, a GUI is presented that queries the user for information about their choice and their situational awareness.

**reschu.game.view.PanelPayloadControls:**

A series of five Components that appear to the right of the visual search task. The optional countdown clock tells how much recommended time remains, the relook button allows a user to leave the current target, the “+” and “-” buttons allow for zooming in and out, and the query button displays the number of pending targets.

**reschu.game.view.PanelTimeLine:**

Displays how much time remains before each vehicle reaches a target or waypoint in its path.

The bottom displays how much total time remains in the mission.