package reschu.app

class reschu.app.AppMain

# Methods:

public AppMain()

* Default constructor

static public void main(String[] args)

* If able to connect, creates a Runnable object with run() that:
  + Creates an AppMain object app
  + Calls app.initDB()
  + calls app.setFrmLogin(), creating the JFrame for user login

private void setFrmLogin()

* Creates and displays a JFrame containing HAL icon, text boxes for username and password, and JComboBox to choose scenario

public void actionPerformed(ActionEvent ev)

* If a scenario is chosen, then the start button is enabled
* If the start button is pressed, then initRESCHU(\_username, \_password) is called

private void initRESCHU(String username, int scenario)

* Creates a Reschu object: new Reschu(GAME\_MODE, scenario, \_username, this, WRITE\_TO\_DATABASE)

public void Restart\_Reschu()

public void TutorialFinished()

private void initDB()

package reschu.constants

class reschu.constants.MyColor

# Variables:

**final** **static** **public** Color *COLOR\_HIGHLIGHT* = **new** Color(255,255,255,255);

**final** **static** **public** Color *COLOR\_VEHICLE* = **new** Color(128,224,255,150);

**final** **static** **public** Color *COLOR\_HIGHLIGHT\_CONTROL*  = **new** Color(128,224,255,255);

**final** **static** **public** Color *COLOR\_HIGHLIGHT\_TAB* = **new** Color(128,224,255,200);

**final** **static** **public** Color *COLOR\_LINE* = **new** Color(128,224,255,150);

**final** **static** **public** Color *COLOR\_VEHICLE\_TIMELINE* = **new** Color(130,148,202,100);

**final** **static** **public** Color *COLOR\_TARGET\_VACANT* = **new** Color(255, 128, 128,150);

**final** **static** **public** Color *COLOR\_TARGET\_OCCUPIED* = **new** Color(240, 10, 10, 150);

**final** **static** **public** Color *COLOR\_TARGET\_DONE* = **new** Color(50,50,50,250);

**final** **static** **public** Color *COLOR\_TARGET\_COMM\_VACANT* = **new** Color(128, 255, 128, 150);

**final** **static** **public** Color *COLOR\_TARGET\_COMM\_OCCUPIED* = **new** Color(128, 200, 128, 150);

**final** **static** **public** Color *COLOR\_TARGET\_INVISIBLE* = **new** Color(130,130,130,150);

**final** **static** **public** Color *COLOR\_TARGET\_INVISIBLE\_OCCUPIED* = **new** Color(50,50,50,150);

**final** **static** **public** Color *COLOR\_VEHICLE\_COMM\_BOUNDARY* = **new** Color(50,250,50,50);

**static** **public** Color *COLOR\_VEHICLE\_PENDING* = **new** Color(128,224,255,150);

package reschu.database

class reschu.database.JDBC

# Methods:

public JDBC() throws Exception

public boolean isConnected()

public ResultSet getResultSet(String str)

public int user\_login(String username, char[] password)

public void disconnectConnection()

private boolean isPasswordCorrect(String correctPassword, char[] userInput)

package reschu.game.controller

class reschu.game.controller.Reschu

# Methods:

public Reschu(int gamemode, int scenario, String username, AppMain main, Boolean database)

* Assigns values to instance variables \_gamemode, \_scenario, \_username, and \_database
* Calls initComponents()

private void initComponents()

* Applies a hardcoded TableLayout to the content pane
* Instantiates a Game object: this.game = new Game(this, \_scenario)
* Instantiates Components displayed on the GUI and puts them in their appropriate places in the table

private void checkIntersect(int vIdx)

* Updates Vehicle’s (of index vIdx-1) knowledge of whether or not its path intersects a hazard

public void vehicle\_location\_changed()

* Repaints map

public void Clock\_Tick(int milliseconds)

public void Game\_Start()

* If in training mode, displays instructions
* Begins execution of Thread(game)
* Enables pnlMap

public void Game\_End()

public void showMesssageOnTopOfMap(String msg, int duration)

public void Rotate\_Clockwise\_Selected()

public void Pan\_Up\_Selected()

public void Rotate\_Counter\_Selected()

public void Pan\_Down\_Selected()

public void Zoom\_In()

public void Zoom\_Out()

public void Submit\_Payload()

public void Vehicle\_Selected\_From\_pnlMap(int idx)

public void Vehicle\_Unselected\_From\_pnlMap()

public void Vehicle\_Engage\_From\_pnlMap(Vehicle v)

public void Payload\_Finished\_From\_pnlPayload(Vehicle v)

public void Payload\_Assigned\_From\_pnlPayload(Vehicle v, Payload p)

public void Payload\_Graphics\_Update()

public void Payload\_Submit(Boolean submit)

public void Vehicle\_Damaged\_By\_Hazard\_Area\_From\_Vehicle(Vehicle v)

public void Target\_Become\_Visible\_From\_Vehicle(Vehicle v)

public void Hide\_Popup(Vehicle v)

public void Vehicle\_Selected\_From\_pnlControl(int idx)

public void Vehicle\_Unselected\_From\_pnlControl()

public void Vehicle\_Goal\_From\_pnlControl(Vehicle v)

public void Vehicle\_WP\_Add\_From\_pnlControl(Vehicle v)

public void Vehicle\_WP\_Del\_From\_pnlControl(Vehicle v)

public void Vehicle\_Engage\_From\_pnlControl(Vehicle v)

private void Engage(Vehicle v)

private void Write(int invoker, int type, int vIdx, String log, int X, int Y)

public void writeConfidenceLevel(String level)

private void play(String arg)

public static void main(String[] args)

many EVT methods to write data

package reschu.game.model

class reschu.game.model.Game

# Methods:

public Game(GUI\_Listener l, int scenario)

* If Reschu.\_database (which is currently hardcoded to be true) is true, then a Thread is created to write to the database
* Instantiates a FrameStart object, passing the variable lsnr to the FrameStart constructor

public synchronized int getElapsedTime()

* Returns this.elapsedTime

public void setListener(GUI\_Listener l)

* this.lsnr = l;

public DBWriter getDBWriter()

* Returns this.dbWriter

private int getSeedNum(int scenario)

private void setProbability\_Target\_Visibility(int scenario)

public void setVehicle(int scenario)

public void setPayload()

private void setMap()

public void run()

* Calls AutoTargetAssignAll(), which assigns unassigned targets to vehicles that did not have targets
* Calls tmr\_clock.start(), starting the countdown to the end of the mission

public Boolean isRunning()

* Returns this.tmr\_clock.isRunning()

public void stop()

public VehicleList getVehicleList()

* Returns this.vehicle\_list, a LinkedList containing all vehicle objects

public PayloadList getPayloadList()

* Returns this.payload\_list

public void vehicle\_location\_change()

public Vehicle Vehicle\_Location\_Check(int x, int y)

public StructSelectedPoint Vehicle\_Goal\_Check(int x, int y)

public StructSelectedPoint Vehicle\_Waypoint\_Check(int x, int y)

public void addScore(int i)

* Increases this.score by i

public int getScore()

* Returns this.score

public Vehicle getCurrentPayloadVehicle()

* Returns this.currentPayloadVehicle

public void setCurrentPayloadVehicle(Vehicle v)

* Assigns v to this.currentPayloadVehicle

public void clearCurrentPayloadVehicle()

* Assigns null to this.currentPayloadVehicle

private void AutoTargetAssignAll()

* Iterates over the vehicles in this.vehicle\_list, passes each to AutoTargetAssign(Vehicle v)
* Has the effect of assigning each vehicle that does not have a target an unassigned target, if one exists

public void AutoTargetAssign (Vehicle v)

* If v does not have a current destination (length of path is zero), and if there is an unassigned target, then v is set to have the coordinates of that target as its goal

public void actionPerformed(ActionEvent e)

public void setTargetUsed(String name, Boolean isUsed)

* For the StructTargetNamePool object obj, in the array this.targetNamePool, such that obj.name == name, isUsed is assigned to obj.used

public String getEmptyTargetName()

* Gets the first unused target name in StructTargetNamePool[] targetNamePool, sets it as used and returns the name

private void setTargetVisibility()

public Boolean getTargetVisibility()

public static double getDistance(double x1, double y1, double x2, double y2)

* Returns the distance between (x1, y1) and (x2, y2)

public static double getDistance(int x1, int y1, int x2, int y2)

* Returns the distance between (x1, y1) and (x2, y2)

class reschu.game.model.Map

### Methods:

Map()

* Default constructor

Map(int width, int height, Game g, GUI\_Listener l)

* Assigns
  + an integer array (width by height) to map\_array
  + g to this.g
  + l to lsnr

public LinkedList<Target> getListAssignedTarget()

* Returns listAssignedTarget

public LinkedList<Target> getListUnassignedTarget()

* Returns listUnassignedTarget

public LinkedList<int[]> getListHazard()

* Returns listHazard

public int getTargetSize(String target\_type)

* Returns the total number of targets (contained in this.listAssignedTarget and this.listUnassignedTarget) that have the attribute target.mission == target\_type

public synchronized void setCellType(int x, int y, int type)

* Assigns type to map\_array[x][y]

public synchronized int getCellType(int x, int y)

* Returns map\_array[x][y]

public synchronized void addTarget(Target t)

* Appends t to the end of listUnassignedTarget

public synchronized void addHazard(int x, int y, int size)

* Appends the element {x, y, size} to the end of listHazard

public synchronized void assignTarget(int[] coordinate)

* Iterates over the list of unassigned targets, and if its position is so close to coordinate, then the target is moved out of the unassigned targets list and put into the assigned targets list
* Represents taking an unassigned target at (or about) the position ‘coordinate’ and designating it as an assigned target

private synchronized void unassignTarget(int[] coordinates)

* Iterates over the list of assigned targets, and if its position is so close to coordinate, then the target is moved out of the assigned targets list and put into the unassigned targets list
* Represents taking an assigned target at (or about) the position ‘coordinate’ and designating it as an unassigned target

public synchronized void delHazard(int idx)

* Removes the idxth element from listHazard

public synchronized boolean isTarget(int x, int y)

* Returns whether or not (x,y) is the position of an unassigned or an assigned target

public synchronized boolean isAssignedTarget(int x, int y)

* Returns whether or not (x,y) is the position of an assigned target

public synchronized boolean isUnassignedTarget(int x, int y)

* Returns whether or not (x,y) is the position of an unassigned target

public synchronized boolean isHazard(int x, int y)

* Returns whether or not (x,y) is the position of a hazard

private boolean boundaryCheck(int x, int y, int[] target\_pos)

* Returns whether or not the point target\_pos is within the square of side length 2w centered at (x,y)
* w equals Math.round(MySize.SIZE\_TARGET\_PXL / MySize.SIZE\_CELL / 2)

public void setHazardArea(Random rnd)

* Adds to listHazard however many more hazards are necessary to make the map contain the appropriate number of hazards for this gameplay mode

private boolean chkOkayToAdd(int x, int y)

* Returns false if there is a target, a hazard or a vehicle within LIMIT\_DISTANCE = 40 of the point (x,y); returns true otherwise

public void delHazardArea(Random rnd, int n)

* Removes n elements at random from listHazard

public void setTargetArea(Random rnd) throws UserDefinedException

* Adds the needed number of land, shore, and comm targets to the map, in appropriate locations with respect to their type
* Throws UserDefined Exception if limit = 100000 locations are searched before all additional targets are determined

private int[] chkTargetOffset(int x, int y)

* Returns {x,y} if (x,y) is 10 pixels away from the edges of the map
* Returns {x±10, y±10} to readjust in whichever dimension is necessary for the point to be 10 pixels away from all borders

public int getAvailableTarget()

* Returns the number of unassigned visible targets

public void garbageTargetCollect()

* If an assigned target is done, then the target’s name is set as unused and the target is removed from the list of assigned targets

private boolean chkOkayToAdd\_TEMPORARY\_FOR\_TUTORIAL\_BY\_CARL(int x, int y)

public void setTargetArea\_TEMPORARY\_FOR\_TUTORIAL\_BY\_CARL(Random rnd)

reschu.game.model.Payload

# Methods:

public Payload(int i, int[] loc, String vType, String tType, String stmt)

* Assigns:
  + i to idx
  + idx + “\_” + vType + “\_” + tType + “.jpg” to filename
  + vType to vehicleType
  + targetType to tType
  + loc to location
  + stmt to statement
  + false to done

reschu.game.model.PayloadList

class reschu.game.model.

StructTargetNamePool

# Methods:

public StructTargetNamePool(String s)

* Assigns:
  + s to this.name
  + false to this.used

public String getName()

* Returns this.name

public synchronized boolean isUsed()

* Returns this.used

public synchronized void setUsed(boolean u)

* Assigns u to this.used

class reschu.game.model.Target

## Methods:

Target()

* Default constructor

Target(String s, int[] p, String m, String t, Boolean v)

* Assigns values to this.name, this.pos, this.mission, this.type, and this.visible, respectively
* Assigns false to this.done

public void setPos(int[] p)

* Assigns p to this.pos

public void setMission(String m)

* Assigns m to this.mission

public void setType(String t)

* Assigns t to this.type

public void setName(String i)

* Assigns I to this.name

public int[] getPos()

* Returns this.pos

public String getMission()

* Returns this.mission

public String getType()

* Returns this.type

public String getName()

* Returns this.name

public void setDone()

* Assigns true to this.done

public boolean isDone()

* Returns this.done

public void setVisible(boolean v)

* Assigns v to this.visible

public boolean isVisible()

* Returns this.visible

static public boolean isTargetType(String s)

* Returns s.equals(MISSION\_LAND) || s.equals(MISSION\_SHORE)
* Represents whether or not the String s equals “LAND” or “SHORE”, the two possible types of targets on the map

class reschu.game.model.Vehicle

# Methods:

public Vehicle(Map m, Game g)

* Initializes:
  + pos\_x, pos\_y to zero
  + this.g to g
  + map to m
  + status to MyGame.STATUS\_VEHICLE\_STATIS
  + vDamage to zero
  + UUV\_stuck\_count to zero
  + UUV\_stuck to false
  + intersect to false

public synchronized void setPos(int x, int y)

* Assigns x and y to pos\_x and pos\_y, respectively

public synchronized LinkedList<int[]> getPath()

* Returns this.path

public synchronized void addPath(int idx, int[] e)

public synchronized void addPathLast(int[] e)

public synchronized void setPath(int idx, int[] e)

public synchronized int getPathSize()

* Return this.path.size()

public synchronized int[] getPathAt(int idx)

public synchronized void removePathAt(int idx)

public synchronized int[] getFirstPath()

public synchronized int[] getLastPath()

public synchronized void removeFirstPath()

public synchronized Map getMap()

public synchronized void setX(int x)

* Assigns x to this.pos\_x

public synchronized int getX()

* Returns this.pos\_x

public synchronized void setY(int y)

* Assigns y to this.pos\_y

public synchronized int getY()

* Returns this.pos\_y

public void setName(String strName)

* Assigns strName to this.name

public String getName()

* Returns this.name

public void setType(String strType)

* Assigns strType to this.type

public String getType()

* Returns this.type

public void setIndex(int idx)

* Assigns idx to this.index

public int getIndex()

* Returns this.index

public void setPayload(String setPayload)

* Assigns setPayload to this.payload

public String getPayload()

* Returns this.payload

public void setVelocity(int milliseconds)

* Assigns milliseconds to this.velocity

public int getVelocity()

* Returns this.velocity

public void setTarget(Target t)

* Assigns t to this.target

public Target getTarget()

* Returns this.target

public void setStatus(int i)

* Assigns i to this.status

public int getStatus()

* Returns this.status

public void setIntersect(boolean b)

* Assigns b to this.intersect

public Boolean getIntersect()

* Returns this.intersect

static public Boolean isVehicleType(String s)

* Returns s.equals(TYPE\_UAV) || s.equals(TYPE\_UUV)

public double getDamage()

* Returns this.vDamage

public void setGuiListener(GUI\_Listener l)

* Assigns l to this.lsnr

private Boolean boundaryCheck(int x, int y, int[] target\_pos)

* Returns whether (x, y) is inside the square of side length 2w centered at target\_pos
* w is Math.round(MySize.SIZE\_TARGET\_PXL / MySize.SIZE\_CELL / 2)

public boolean isAssignededTarget(int x, inty)

public void addGoal(int x, int y)

public void changeGoal(int[] ex\_goal, int x, int y)

public synchronized int addWaypoint(int x, int y)

public synchronized int addWaypoint(int x, int y, int idx)

public void delWaypoint(int x, int y)

public void delWaypoint(int[] coordinate)

public void changeWaypoint(int ex\_x, int ex\_y, int new\_x, int new\_y)

public boolean hasGoal()

public Boolean hasWaypoint()

# Moving algorithms:

public double getDistance(int pos\_x, int pos\_y)

public void moveRandom(int i)

public void moveHillClimbing()

public void moveBestFirst()

public void moveTo(int direction)

private void payloadCheck(int pos\_x, int pos\_y)

public synchronized Boolean chkValidMove(int direction)

public synchronized Boolean chkValidPosition(int width, int height)

public void chkHazardArea()

private void updateVisibility(int x, int y)

public void COM\_Payload()

package reschu.game.utils

class

package reschu.game.view

class reschu.game.view.FrameStart

# Methods:

public FrameSart(GUI\_Listener l)

* Creates a JFrame containing the HAL logo, a progress bar and a start button
* The start button is initially disabled

public void actionPerformed(ActionEvent e)

* ActionEvent can come from either Timer or JButton
* The progress bar is updated, and if progress is complete, the start button is enabled
* If the start button is clicked, then this frame becomes invisible and the simulation begins by calling lsnr.Game\_Start()

class reschu.game.view.PanelLikertScale

## extends JPanel implements ActionListener

# Methods:

public PanelLikertScale(GUI\_Listener game)

* Assigns game to this.reschu
* Places a prompt, radio buttons, and a submit “OK” button on the JPanel

public void setPopup(Popup p)

* Assigns p to this.pop

public void setCorrect(boolean c)

* Assigns c to this.correct

public void actionPerformed(ActionEvent e)

* Passes the user’s choice as a String to this.reschu.writeConfidenceLevel(String level)
* Hides this.pop if its value is not null
* Passes a String to PanelMsgBoard.Msg(String msg) telling the user whether or not the choice is correct

class reschu.game.view.PanelMsgBoard

# Methods:

public PanelMsgBoard()

public static void Msg(String msg)

private static String Now()

private void insert\_grid(GridBagConstraints gbc, Component cmpt, int x, int y, int width, int height, double percent\_x, double percent\_y, int ins)

public void actionPerformed(ActionEvent e)

class reschu.game.view.PanelPayload

# Methods:

public PanelPayload(GUI\_Listener e, String strTitle, GLJPanel payload\_canvas, Game g)

private void initTextRenenders()

private void makeVibrateThread()

public void init(GLAutoDrawable drawable)

public void display(GLAutoDrawable drawable)

public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height)

public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, boolean deviceChanged)

private void initAnimRenderer()

private void updateAnimRenderer()

public synchronized boolean isEnabled()

private synchronized void glEnabled(boolean b)

public void setClicked(boolean c)

public void reset\_variables()

public void set\_payload(Vehicle v)

private void displayText(GLAutoDrawable drawable)

private void displayAnimRenderer(GLAutoDrawable drawable, int viewport\_x, int viewport\_y, float x\_off, float y\_off)

private void camera\_pers(GL gl)

private void unproj(GL gl, int x, int y, double[] wcoord)

public void mouse\_click(java.awt.event.MouseEvent m\_ev)

private void setCorrect()

* If chosen location is within a square of side length 2\*offset (offset =150) pixels centered at the actual location (wcoord[]), then this.correct is set to be true, otherwise, it is set to be false

public void checkCorrect()

//Payload control camera:

public double getRotating()

public void setRotating(double alpha)

public float getPanX()

public void setPanX(float alpha)

public float getPanY()

public void setPanY(float alpha)

public double getZoom()

public void setZoom(double alpha)

public void r\_c\_2()

public void r\_c\_c\_2()

public void pan(float x, float y, int time)

public void zoom\_in()

public void zoom\_out()

public JPopupMenu getPopMenu()

private void setPopup()

* If PanelPayload.USE\_POPUP is true:
  + Submit and cancel items are added to popMenu
* If PanelPayload.USE\_POPUP is false (currently set to false):
  + Submit and cancel buttons are assigned to btnSubmit and btnCancel, respectively
* When “Submit” is clicked, the answer is checked
* When either is clicked, both “Submit” and “Cancel” are removed from the screen

private void showPopup(Component invoker, int x, int y)

* If PanelPayload.USE\_POPUP is true:
  + this.popMenu is placed on the screen at (x,y)
* If PanelPayload.USE\_POPUP is false (currently set to false):
  + btnSubmit and btnCancel are placed on the screen near (x,y)

private void hidePopup()

* If PanelPayload.USE\_POPUP is true:
  + this.popMenu is set as invisible
* If PanelPayload.USE\_POPUP is false (currently set to false):
  + btnSubmit and btnCancel are removed from the screen

package reschu.tutorial

class

package reschu.userinput

class