CarBot_Dalisay: The Robot with Nine Lives

Enriquez, Deangelo M.

University of the Philippines - Baguio

CarBot_Dalisay: The Robot with Nine Lives

From the name itself, `CarBot_Dalisay` was inspired by *Ricardo "Cardo" Dalisay*, the protagonist of the long-time Philippine action series *FPJ's Ang Probinsyano*. In context, *Cardo* was a former policeman who pursues justice against the tyranny of his country and attacks those that gets in his way. He was famously dubbed on the web as an 'immortal' character, which began by his inevitable plot armor and his ability to survive in dire situations. Similarly, `CarBot_Dalisay` performed the identical way as *Cardo* in a battlefield situation. In statistical terms, `CarBot_Dalisay` 's survivability rate against the sample robots sat at 80%, which idiomatically labeled `CarBot_Dalisay` as a 'robot with nine lives.' `CarBot_Dalisay` furiously traveled randomly and unpredictably, presenting its opponents a tricky time accurately shooting at `CarBot_Dalisay` 's exact location.

Generally, the main feature `CarBot_Dalisay` was its firing strategy called 'linear targeting', a method wherein it supposes that the enemy will advance to maneuver at the identical speed and direction (RoboWiki, 2012). During this case, 'linear targeting' was modified in a way that the closest approaching opponent was automatically locked as a target. Furthermore, `CarBot_Dalisay` will only proceed to fire if the opponent's gun is in a cooldown status. `CarBot_Dalisay` also utilized the concept of energy management, wherein it fires only if the enemy's distance is less than 600 pixels away from `CarBot_Dalisay`. Consequently, its firing power was directly proportional to the opponent's distance, which denotes that opponents that ram into `CarBot_Dalisay` will receive a greater amount of damage.

Code Structure

`CarBot_Dalisay`'s behavior pattern is mainly based from Miked0801's `DustBunny`. Major improvements were done such as energy management and modifying its linear targeting.

Classes

`Robot` – used as a structure to create a compatible robot for RoboCode

`Color` – used to customize the colors of the robot

`ScannedRobotEvent` – sent to `OnScannedRobot()` when scanning a robot

`HitWallEvent` – sent to `onHitWall()` when colliding a wall

`RobotDeathEvent` – sent to `onRobotDeath()` when an opponent robot dies

Inherited Class

`AdvancedRobot` – The application of inheritance to build RoboCode is crucial because it allowed similar but distinct objects to exhibit their common capabilities without requiring the code to be in two places (TutorialsPoint, 2021). The class `AdvancedRobot` was extended to provide `CarBot_Dalisay` all of the methods defined within the RoboCode API. `AdvancedRobot` permits non-blocking call, custom events, and provides methods that are suitable in building a robust robot.

Main Variables

Global variables:

- `static double galaw` – sets up how far the robot will move inside the battlefield. Within the `run()` method, `galaw` is supplied with `Double.MAX_VALUE`, which was the utmost value a double can have. This value was used to enable `CarBot_Dalisay` to maneuver randomly and unpredictably.

- `static double distansya` – Specifically, `distansya` provides the robot information on the distance of the closest scanned robot.
- `static String kalaban` – provides the name of what robot will `CarBot_Dalisay` lock its target into.

`double kalabanDistansya` – Inside the `onScannedRobot` method, `kalabanDistansya` was used to store `e.getDistance()`, a method that returns the distance of the robot. To not confuse with the variable `distansya`, `kalabanDistansya` was provided to lock onto a new target if an opponent was closer than the current one. After locking to a new target, `kalabanDistansya` will gather the exact position and angle of the target, which is important to set up the 'linear targeting'.

Methods

`private Color Kulay(int r, int g, int b)` – Inherited from the `Color` class, `Kulay` provides a wider range of colors for the body, gun, and radar of the robot.

Inherited from `Robot` (in order of usage):

- `public void run()`
- `setAdjustGunForRobotTurn()` – When true, it sets the gun to turn independent from the robot's turn. This is vital for the robot such that it can hit stationary targets.
- `onRobotDeath()` – This is called when an opponent robot dies. Inside `run()`, `onRobotDeath()` was set to `null` to search out a new target. On the other hand, `onRobotDeath()` was also used inside `onScannedRobot()` to reset the distance.

- `turnRadarRightRadians()` – Immediately turns the robot's radar to the right by radians. Applying the programming rule of thumb, `Double.MAX_VALUE` was used instead of the `while()` loop to save space.
- `public void onScannedRobot(ScannedRobotEvent e)` – called when `CarBot_Dalisay` sees another robot. This was where roughly 90 percent of `CarBot_Dalisay`'s behavior was coded.
- `getName()` – returns the robot's name. If this method is equal to the variable `kalaban`, then `CarBot_Dalisay` will point out to this target and fire when the target's gun is in cooldown.
- `getGunHeat()` – returns the current heat of the gun. A value of zero indicates that the robot is unable to fire.
- `public void onHitWall()` – Upon hitting a wall, a condition was applied such that the value of `galaw` will be set and reduced to a value of 200. This was executed to maintain `CarBot_Dalisay` away from the center.
- `public void onRobotDeath()` – called when an opponent robot dies. After the death of an opponent, distansya will reset its distance to `Double.MAX_VALUE` to avoid getting stuck at aiming within the same place.

Inherited from `AdvancedRobot` (in order of usage):

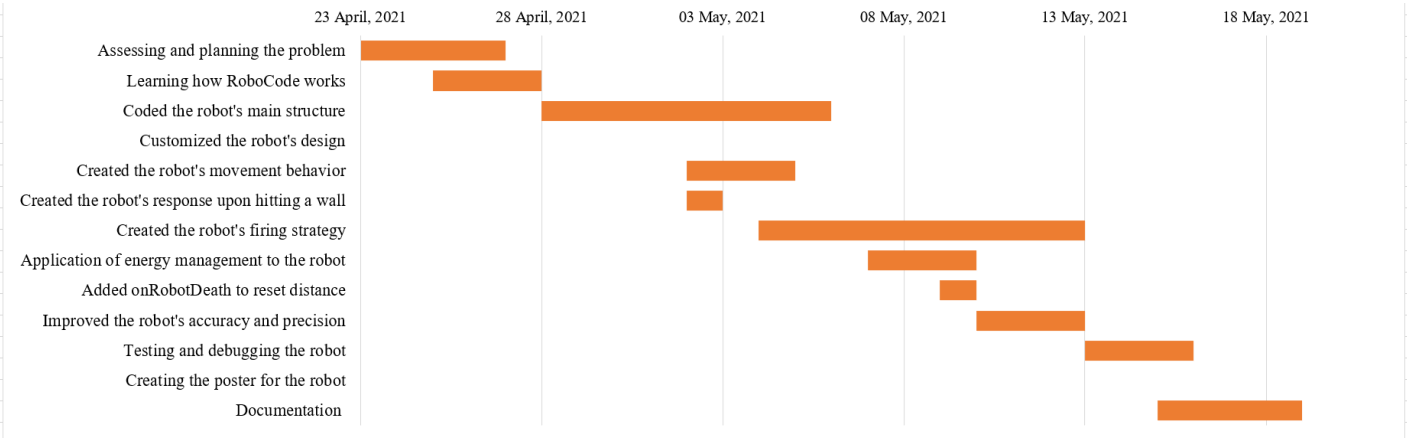
- `getDistanceRemaining()` – returns the distance remaining in the robot's current move measured in pixels. This was used inside the `ScannedRobotEvent` so that when the value of `getDistanceRemaining()` hits zero or stationary, the robot will not

continue to stop or ram its opponent, but instead will turn away and move in an opposite direction.

- `setAhead()` – sets the robot ahead to maneuver ahead, but in this robot, it was used to move ahead backward. This was simply because `setAhead()` was used inside the case when the `getDistanceRemaining()` is equal to the value of zero.
- `setTurnRightRadians()` – sets the robot's body to turn right by radians. Therefore, the value of 90 only defines that the robot's body will turn at a right angle.
- `getGunTurnRemaining()` – returns the angle remaining within the gun's turn, in degrees. This was added to `CarBot_Dalisay` to repair the problem of linear targeting's imprecise aim.
- `setFireBullet()` – sets the gun to fire a bullet when the subsequent execution takes place. Inside this method, `CarBot_Dalisay`'s energy was managed through this formula: `getEnergy() * 15 / kalabanDistansya`. Assuming that the value used to determine a robot's fire power is fifteen, the formula demonstrates that the nearer the opponent is to the robot, the higher the amount of fire power will be manifested. On the other hand, a lower amount of firepower is going to be manifested against the farther opponent, so that it will have enough time to reach and hit the opponent. Furthermore, the relationship between the robot's firepower and its energy is directly proportional to each other.

- `setTurnGunRightRadians()` – sets the robot's gun to turn right by radians. Inside this method, it shows the code for 'linear targeting' with some major improvements such as fixing its aim and adjusting the energy as the distance of the opponent from the robot increases. If the location of the opponent is beyond 600 pixels, then the robot will not shoot this opponent to conserve its energy.

Timeline



References

AdvancedRobot (Robocode 1.9.4.2 API). (2021, May 10). Robocode. https://robocode.sourceforge.io/docs/robocode/robocode/AdvancedRobot.html

DustBunny - Robowiki. (2017, August 2). RoboWiki. https://robowiki.net/wiki/DustBunny

Java - Inheritance - Tutorialspoint. (n.d.). Tutorialspoint. https://www.tutorialspoint.com/java/java_inheritance.htm

Linear Targeting - Robowiki. (2012, May 12). Robowiki. https://robowiki.net/wiki/Linear_Targeting

Robot (Robocode 1.9.4.2 API). (2021, May 10). Robocode. https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html

Wikipedia contributors. (2021, May 14). *Ang Probinsyano*. Wikipedia. https://en.wikipedia.org/wiki/Ang_Probinsyano