Team Godspeed

Generated by Doxygen 1.8.20

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 godspeed Namespace Reference

The primary namespace for the project.

Namespaces

- actions

    A namespace containing actions for use in autonomous mode.
- Bauble

    A namespace containing the state machine and driver control code for the Bauble robot.
- behaviors

    A namespace containing autonomous mode behavior bindings.
- Binder

    A namespace containing functions for binding value producing functions to value consuming functions.
- framework

    Contains core classes and functions used as building blocks for the rest of the project.
- inputs

    Contains all classes and namespaces for input devices.
- ModeControl

    A namespace for controlling the operation of the robot.
- outputs

    Contains all classes and namespaces for output devices.
- StateMachine

    A namespace for executing a state machine.
- Tchotchke

    A namespace containing the state machine and driver control code for the Tchotchke robot.

Classes

- class Binding

    A class that represents a binding between two functions.
- class Debouncer

    A class for debouncing boolean data using a delay.
- class State

    A state class used to create statemachines to execute using the StateMachine namespace.
- class WinAvg

    A class to apply windowed averaging.
- class WinMin

    A class to apply windowed minimum.

3.1.1 Detailed Description

The primary namespace for the project.

3.2 godspeed::actions Namespace Reference

A namespace containing actions for use in autonomous mode.

Enumerations

- enum mode {
  ON, **OFF**, **LEFT**, **RIGHT**,
  FORWARD, **REVERSE**, **BACKWARD** }

  Operation modes for output devices.

Functions

- void BallCollectors (mode m)

  Sets ball collector mode.
- void BallScorer (mode m)

  Sets ball scorer mode.
- void Drive (mode m)

  Sets drivetrain movement mode.
- void Drive (mode m, int dur)

  Sets drivetrain movement mode for a time interval.
- void Turn (mode m)

  Sets drivetrain turning mode.
- void Turn (mode m, int dur)

  Sets drivetrain turning mode for a time interval.

3.2.1 Detailed Description

A namespace containing actions for use in autonomous mode.

3.3 godspeed::Bauble Namespace Reference

A namespace containing the state machine and driver control code for the Bauble robot.

Functions

- void ent1 ()

  Entry action for state 1.
- void ent2 ()

  Entry action for state 2.
- void ent3 ()

  Entry action for state 3.
- void ent4 ()

  Entry action for state 4.
- void ent5 ()

  Entry action for state 5.
- void StartAutonomous ()

  Start Bauble autonomous routine.
- double **expander_pos** ()
- Binding **ExpanderBinding** (expander_pos, outputs::BallScorer::ExpanderPosition)
- Binding **DCtrl_XSpeed** (RemoteController::LeftStickX, OmniDrive3Wheel::XSpeed)
- Binding **DCtrl_YSpeed** (RemoteController::LeftStickY, OmniDrive3Wheel::YSpeed)
- Binding **DCtrl_AngleSpeed** (RemoteController::RightStickX, OmniDrive3Wheel::AngleSpeed)
- Binding **DCtrl_CenterTread** (RemoteController::LeftTrigger, BallScorer::TreadSpeed)
- Binding **DCtrl_CollectorTreads** (RemoteController::RightTrigger, BallCollector::TreadSpeed)
- Binding **DCtrl_EnableSaturation** (RemoteController::AButton, OmniDrive3Wheel::EnableSaturation)
- void BindDriverControl ()

  Bind Bauble driver control.
- void UnBindDriverControl ()

  Un-bind Bauble driver control.

3.3.1 Detailed Description

A namespace containing the state machine and driver control code for the Bauble robot.

3.4 godspeed::behaviors Namespace Reference

A namespace containing autonomous mode behavior bindings.

Functions

- double AlignPipe (double d)

  A pipe function used in the aligning behaviors.
- double Stop ()

  A source function used in some behaviors.
- double LocateSpeed ()

  A source function used in locate object behavior.
- double TurnLeftSpeed ()

  A source function used in turn left behavior.
- double TurnRightSpeed ()

  A source function used in turn right behavior.
- double BallScorerSpeed ()

A source function used in the score ball behavior.
- double BallPickupSpeed ()

  A source function used in the ball pickup behavior.
- double ForwardSpeed ()

  A source function used in the move forward behavior.
- double BackwardSpeed ()

  A source function used in the move backward behavior.
- double RightSpeed ()

  A source function used in the move forward behavior.
- double LeftSpeed ()

  A source function used in the move backward behavior.
- Binding AlignWithBall (inputs::VisionSensor::BallXOffset, AlignPipe, outputs::OmniDrive3Wheel::AngleSpeed)

  A behavior that turns the robot to face the largest ball found.
- Binding AlignWithGoal (inputs::VisionSensor::GoalXOffset, AlignPipe, outputs::OmniDrive3Wheel::AngleSpeed)

  A behavior that turns the robot to face the largest goal backboard icon found.
- Binding MoveForward (ForwardSpeed, outputs::OmniDrive3Wheel::YSpeed)

  A behavior that moves the robot forward.
- Binding MoveBackward (BackwardSpeed, outputs::OmniDrive3Wheel::YSpeed)

  A behavior that moves the robot forward.
- Binding MoveRight (RightSpeed, outputs::OmniDrive3Wheel::XSpeed)

  A behavior that moves the robot forward.
- Binding MoveLeft (LeftSpeed, outputs::OmniDrive3Wheel::XSpeed)

  A behavior that moves the robot forward.
- Binding TurnLeft (TurnLeftSpeed, outputs::OmniDrive3Wheel::AngleSpeed)

  A behavior that turns the robot left.
- Binding TurnRight (TurnRightSpeed, outputs::OmniDrive3Wheel::AngleSpeed)

  A behavior that turns the robot right.
- Binding LocateObject (LocateSpeed, outputs::OmniDrive3Wheel::AngleSpeed)

  A behavior that turns the robot in place in order to locate an object.
- Binding ScoreBall (BallScorerSpeed, outputs::BallScorer::TreadSpeed)

  A behavior that turns the center tread to move a ball into the goal.
- Binding PickUpBall (BallPickupSpeed, outputs::BallCollector::TreadSpeed)

  A behavior that turns the ball collector treads in order to pickup a ball.
- Binding AvoidObstacle (inputs::RangeFinders::Nearness, outputs::OmniDrive3Wheel::XSpeed)

  A behavior to move the robot away from a detected obstacle.
- Binding StopX (Stop, outputs::OmniDrive3Wheel::XSpeed)

  A behavior to stop the robot's x-direction velocity.
- Binding StopY (Stop, outputs::OmniDrive3Wheel::YSpeed)

  A behavior to stop the robot's y-direction velocity.
- Binding StopAngle (Stop, outputs::OmniDrive3Wheel::AngleSpeed)

  A behavior to stop the robot's angular velocity.
- Binding StopCollectors (Stop, outputs::BallCollector::TreadSpeed)

  A behavior to stop the ball collector treads.
- Binding StopScorer (Stop, outputs::BallScorer::TreadSpeed)

  A behavior to stop the ball scorer tread.

Variables

- double **AlignAgression** = 1

3.4.1 Detailed Description

A namespace containing autonomous mode behavior bindings.

3.5 godspeed::Binder Namespace Reference

A namespace containing functions for binding value producing functions to value consuming functions.

Functions

- void AddBinding (Binding &b)

  Add a binding to the binder.
- void SetBindings (std::list< Binding ∗ > &l)

  Set the bindings list (overwrites current binding list)
- void ClearBindings ()

  Remove all current bindings.
- void RemoveBinding (Binding &b)

  Remove given binding from the update list.
- void Update ()

  Iterates through and updates all bindings. This does NOT need to be called manually.
- void Kill ()

  Stops binder thread.
- void Init ()

  Runs the binder update function on it's own thread.

Variables

- std::list< Binding ∗ > bindings = std::list<Binding∗>()

  A list of bindings.
- thread ∗ **tptr**

3.5.1 Detailed Description

A namespace containing functions for binding value producing functions to value consuming functions.

There are 3 types of functions considered here, source functions which take no values and return a double, pipe functions which take a double and return a double, and sink functions which take a double and return nothing. Once the Binder is initialized, it will go through and repeatedly call the sink functions with the value produced by the source functions as the input.

3.6 godspeed::framework Namespace Reference

Contains core classes and functions used as building blocks for the rest of the project.

3.6.1 Detailed Description

Contains core classes and functions used as building blocks for the rest of the project.

3.7 godspeed::inputs Namespace Reference

Contains all classes and namespaces for input devices.

Namespaces

- BallStorage

  A namespace with a function for accessing the current number of balls stored in the robot.
- RangeFinders

  A namespace containing functions to access the distance values from all the rangefinders.
- RemoteController

  A namespace containing functions for accessing all remote controller inputs.
- VisionSensor

  A namespace containing functions for accessing data from the Vision Sensor.

Classes

- class PathScript

  A class for creating scripted paths for the robot to take.

3.7.1 Detailed Description

Contains all classes and namespaces for input devices.

3.8 godspeed::inputs::BallStorage Namespace Reference

A namespace with a function for accessing the current number of balls stored in the robot.

Functions

- Debouncer **inc_debounce** (1000)
- Debouncer **dec_debounce** (1000)
- void inc ()

  Increment the ball count. This does NOT need to be called manually.
- void dec ()

  Decrement the ball count. This does NOT need to be called manually.
- double BallCount ()

  Returns the current ball count.
- void Init ()

  Setup the ball storage counter to track balls.

Variables

- int BallCounter

  A variable for tracking the ball count.

3.8.1 Detailed Description

A namespace with a function for accessing the current number of balls stored in the robot.

The Init() function must be called before the ball storage tracking can begin. That function attaches the inc and dec function as callbacks for when bumper A and bumper B are pressed (respectively).

3.9 godspeed::inputs::RangeFinders Namespace Reference

A namespace containing functions to access the distance values from all the rangefinders.

Functions

- double LeftDistance ()

  returns the distance value of Left Range Finder
- double RightDistance ()

  returns the distance value of Right Range Finder
- double Nearness ()

  returns a value between -1 and 1 representing how close an obstacle is to the right or left side of the robot respectively

Variables

- double **leftDistVar** = 0
- double **rightDistVar** = 0
- double **nearThreshold** = 500

3.9.1 Detailed Description

A namespace containing functions to access the distance values from all the rangefinders.

3.10 godspeed::inputs::RemoteController Namespace Reference

A namespace containing functions for accessing all remote controller inputs.

Functions

- double **UpButton** ()
- double **DownButton** ()
- double **RightButton** ()
- double **LeftButton** ()
- double **XButton** ()
- double **YButton** ()
- double **AButton** ()
- double **BButton** ()
- double **RightTrigger** ()
- double **RightBumper** ()
- double **LeftTrigger** ()
- double **LeftBumper** ()
- double **LeftStickX** ()
- double **LeftStickY** ()
- double **RightStickX** ()
- double **RightStickY** ()

3.10.1 Detailed Description

A namespace containing functions for accessing all remote controller inputs.

3.11 godspeed::inputs::VisionSensor Namespace Reference

A namespace containing functions for accessing data from the Vision Sensor.

Functions

- WinMin **BallDistVar** (50)
- WinMin **GoalDistVar** (50)
- WinAvg **BallCountVar** (10)
- WinAvg **GoalCountVar** (10)
- void Init ()

  Initialization for vision sensor, fills ball and goal distance variables with infinity.
- double CalculateDistance (double obj_w, double obj_h, double obj_x, double obj_w_actual, double obj_h_↩ actual)

  Calculated distance to an object, given it's dimensions etc.
- double GetDistance (signature sig, double obj_w, double obj_h)

  Calculated distance to the largest object given it's signature and actual width and height.
- double XOffset ()

  Get the X offset of the largest object from the center of the screen, normalize to between -1 and 1.
- double YOffset ()

  Get the Y offset of the largest object from the center of the screen, normalize to between -1 and 1.
- double Size ()

  Returns the width, in pixels, of the largest object.
- double BallDistance ()

  Returns the distance, in inches, to the largest ball found [NOT IMPLEMENTED].
- void BallDistanceScan ()

Fills ball distance variable window with data.
- double BallXOffset ()

  Returns the X offset of the largest ball from the center of the screen, normalize to between -1 and 1.
- double BallYOffset ()

  Returns the Y offset of the largest ball from the center of the screen, normalize to between -1 and 1.
- double BallCount ()

  Returns the number of balls found.
- double GoalCount ()

  Returns the number of goals found.
- double GoalDistance ()

  Returns the distance, in inches, to the largest goal backboard icon found [NOT IMPLEMENTED].
- void GoalDistanceScan ()

  Fills goal distance variable window with data.
- double GoalXOffset ()

  Returns the X offset of the largest goal backboard icon from the center of the screen, normalize to between -1 and 1.
- double GoalYOffset ()

  Returns the Y offset of the largest goal backboard icon from the center of the screen, normalize to between -1 and 1.

Variables

- double ScreenWidth = 314

  Width of the screen in pixels.
- double ScreenHeight = 210

  Height of the screen in pixels.
- double VerticalFOV = 20.3341 $*$ (3.14/180)

  Vertical field of view in radians.
- double HorizontalFOV = 30.2145 $*$ (3.14/180)

  Horizontal field of view in radians.
- double XOffsetFudge = 0

  Number added to X-Offset for tuning.
- double BallWidth = 6.3

  Ball width in inches.
- double BallHeight = 6.3

  Ball height in inches.
- double BackboardWidth = 4.375

  Goal backboard icon width in inches.
- double BackboardHeight = 6.1875

  Goal backboard icon height in inches.
- signature $*$ BallSig = &Vision20__RED_BALL

  Signature for ball (changes depending on team color)

3.11.1 Detailed Description

A namespace containing functions for accessing data from the Vision Sensor.

3.11.2 Function Documentation

3.11.2.1 BallCount()

```
double godspeed::inputs::VisionSensor::BallCount ( )
```

Returns the number of balls found.

Take a snapshot looking for the ball

3.11.2.2 BallDistance()

```
double godspeed::inputs::VisionSensor::BallDistance ( )
```

Returns the distance, in inches, to the largest ball found [NOT IMPLEMENTED].

Take a snapshot looking for the ball, and then return Distance()

3.11.2.3 BallXOffset()

```
double godspeed::inputs::VisionSensor::BallXOffset ( )
```

Returns the X offset of the largest ball from the center of the screen, normalize to between -1 and 1.

Take a snapshot looking for the ball, and then return XOffset()

3.11.2.4 BallYOffset()

```
double godspeed::inputs::VisionSensor::BallYOffset ( )
```

Returns the Y offset of the largest ball from the center of the screen, normalize to between -1 and 1.

Take a snapshot looking for the ball, and then return YOffset()

3.11.2.5 GoalCount()

```
double godspeed::inputs::VisionSensor::GoalCount ( )
```

Returns the number of goals found.

Take a snapshot looking for the goal

3.11.2.6 GoalDistance()

```
double godspeed::inputs::VisionSensor::GoalDistance ( )
```

Returns the distance, in inches, to the largest goal backboard icon found [NOT IMPLEMENTED].

Take a snapshot looking for the goal, and then return Distance()

3.11.2.7 GoalXOffset()

```
double godspeed::inputs::VisionSensor::GoalXOffset ( )
```

Returns the X offset of the largest goal backboard icon from the center of the screen, normalize to between -1 and 1.

Take a snapshot looking for the goal, and then return XOffset()

3.11.2.8 GoalYOffset()

```
double godspeed::inputs::VisionSensor::GoalYOffset ( )
```

Returns the Y offset of the largest goal backboard icon from the center of the screen, normalize to between -1 and 1.

Take a snapshot looking for the goal, and then return YOffset()

3.12 godspeed::ModeControl Namespace Reference

A namespace for controlling the operation of the robot.

Enumerations

- enum **robot** { **Robot_Bauble**, **Robot_Tchotchke** }
- enum **team** { **Team_Red**, **Team_Blue** }
- enum **mode** { **Autonomous_Mode**, **Driver_Control_Mode**, **Competition_Mode** }

Functions

- void StartAutonomous ()

  Starts autonomous mode.
- void StartDriverControl ()

  Starts driver control mode.
- void EndAutonomous ()

  Disables autonomous mode.
- void EndDriverControl ()

  Disables driver control mode.
- void StartCompetition ()

  Starts competition watcher to watch for field control events and change mode appropriately.
- void **StartCompetitionTest** ()
- void StopRobot ()

  Stops the robots movement.
- void Init ()

  Start the mode controller (set Robot, Team, and Mode before calling this)
- void CompetitionWatcher ()

  Thread function of competition watcher thread.

Variables

- robot Robot

  Set this depending on the robot being used (Robot_Tchotchke or Robot_Bauble)
- team Team

  Set this depending on the team color (Team_Red, Team_Blue)
- mode Mode

  Set this depending on the desired operation mode (Autonomous_Mode, Driver_Control_Mode, Competition_Mode)
- competition **Competition**

3.12.1 Detailed Description

A namespace for controlling the operation of the robot.

3.13 godspeed::outputs Namespace Reference

Contains all classes and namespaces for output devices.

Namespaces

- BallCollector

  A namespace containing functions for controlling the ball collector.
- BallScorer

  A namespace containing functions for controlling the ball scorer.
- OmniDrive3Wheel

  A namespace with functions for controlling the drive train.
- OutputUtilities

  A namespace for utility functions used by output classes.

3.13.1 Detailed Description

Contains all classes and namespaces for output devices.

3.14 godspeed::outputs::BallCollector Namespace Reference

A namespace containing functions for controlling the ball collector.

Functions

- void TreadSpeed (double speed)

  Sets the speed of the two collector treads.
- void Stop ()

  Stops ball collector treads.

3.14.1 Detailed Description

A namespace containing functions for controlling the ball collector.

This namespace contains a single function that corresponds to both collector arm treads.

3.15 godspeed::outputs::BallScorer Namespace Reference

A namespace containing functions for controlling the ball scorer.

Functions

- void Init ()

  Initialization for ball scorer, zeroes out position of left and right ball guide motors.
- void TreadSpeed (double speed)

  Sets the speed of the center tread.
- void SpinLeftExpander ()

  Spins the left expander motor to the value stored in expanderVar.
- void SpinRightExpander ()

  Spins the right expander motor to the value stored in expanderVar.
- void ExpanderPosition (double angleDeg)

  Sets the angular position of the motor that extends the ball guide.
- void Stop ()

  Stops ball scorer tread.

Variables

- double **expanderVar**

3.15.1 Detailed Description

A namespace containing functions for controlling the ball scorer.

This class contains two functions, one for the center tread and one for the ball guide expander.

3.16 godspeed::outputs::OmniDrive3Wheel Namespace Reference

A namespace with functions for controlling the drive train.

Functions

- WinAvg XSpeedVar (150)

  An object to track the current desired x-speed and apply smoothing to prevent jerky movement.
- WinAvg YSpeedVar (150)

  An object to track the current desired y-speed and apply smoothing to prevent jerky movement.
- void SetVelocity (double x, double y, double a)

  Set the X, Y, and angular velocities of the drivetrain.
- void XSpeed (double x)

  Sets the x-speed of the drivetrain.
- void YSpeed (double y)

  Sets the y-speed of the drivetrain.
- void AngleSpeed (double a)

  Sets the angular speed of the drivetrain.
- void EnableSaturation (double e)

  Allows inputs to motors to saturate, will cause strange diagonal movement but allows for maximum orthogonal speed.
- void Stop ()

  Stops all drivetrain motors.

Variables

- double AngleSpeedVar

  A variable to track the current desired angular speed.
- bool **SaturationEnabled** = false

3.16.1 Detailed Description

A namespace with functions for controlling the drive train.

3.17 godspeed::outputs::OutputUtilities Namespace Reference

A namespace for utility functions used by output classes.

Functions

- void setMotorSpeed (double motorSpeed, motor &m)

  Sets the speed of a motor.

3.17.1 Detailed Description

A namespace for utility functions used by output classes.

3.17.2 Function Documentation

3.17.2.1 setMotorSpeed()

```
void godspeed::outputs::OutputUtilities::setMotorSpeed (
            double motorSpeed,
            motor & m )
```

Sets the speed of a motor.

Takes a value -1 to 1 and sets the velocity appropriately. Has a "dead zone" of -0.1 to 0.1 which it treats as equal to zero (no movement)

3.18 godspeed::StateMachine Namespace Reference

A namespace for executing a state machine.

Functions

- void ChangeState (State &state)

  Changes the state of the state machine, performs entry and exit actions and handles activity enabling and disabling.
- void Update ()

  Update function called by the State Machine thread.
- void Start (State &starting_state)

  Starts executing a state machine given the initial state.
- void Kill ()

  A function to stop the state machine thread.
- void Init ()

  Runs the state machine update function on it's own thread.

Variables

- State ∗ **currentState**
- thread ∗ **tptr**
- bool **kill** = false

3.18.1 Detailed Description

A namespace for executing a state machine.

3.19 godspeed::Tchotchke Namespace Reference

A namespace containing the state machine and driver control code for the Tchotchke robot.

Functions

- bool CenterLineBallNotFound ()

  Condition for if the center line ball is close enough that it probably hasn't been moved by opponent robots or anything.
- void ent1 ()

  Entry action for state 1.
- void ent2 ()

  Entry action for state 2.
- void ent3 ()

  Entry action for state 3.
- void ent5 ()

  Entry action for state 5.
- void ent7 ()

  Entry action for state 7.
- void ent8 ()

  Entry action for state 8.
- void StartAutonomous ()

  Start Tchotchke autonomous routine.
- Binding **DCtrl_XSpeed** (RemoteController::LeftStickX, OmniDrive3Wheel::XSpeed)
- Binding **DCtrl_YSpeed** (RemoteController::LeftStickY, OmniDrive3Wheel::YSpeed)
- Binding **DCtrl_AngleSpeed** (RemoteController::RightStickX, OmniDrive3Wheel::AngleSpeed)
- Binding **DCtrl_CenterTread** (RemoteController::LeftTrigger, BallScorer::TreadSpeed)
- Binding **DCtrl_CollectorTreads** (RemoteController::RightTrigger, BallCollector::TreadSpeed)
- Binding **DCtrl_EnableSaturation** (RemoteController::AButton, OmniDrive3Wheel::EnableSaturation)
- void BindDriverControl ()

  Bind Tchotchke driver control.
- void UnBindDriverControl ()

  Un-bind Tchotchke driver control.
- double **expander_pos** ()
- Binding **ExpanderBinding** (expander_pos, outputs::BallScorer::ExpanderPosition)

3.19.1 Detailed Description

A namespace containing the state machine and driver control code for the Tchotchke robot.

Chapter 4

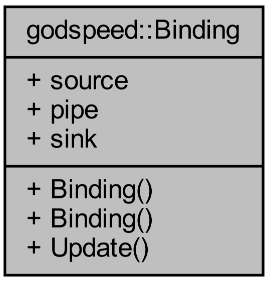
Class Documentation

4.1 godspeed::Binding Class Reference

A class that represents a binding between two functions.

```
#include <binder.h>
```

Collaboration diagram for godspeed::Binding:



Public Member Functions

- Binding (double(∗src)(void), double(∗pip)(double), void(∗snk)(double))

  Constructor that accepts a source, pipe, and sink function.
- Binding (double(∗src)(void), void(∗snk)(double))

  Constructor that accepts a source and sink function.
- void Update ()

  Update the binding.

Public Attributes

- double(∗ **source**)(void)
- double(∗ **pipe**)(double)
- void(∗ **sink**)(double)

4.1.1 Detailed Description

A class that represents a binding between two functions.

The documentation for this class was generated from the following file:

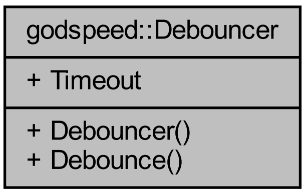
- C:/Users/anzel/source/repos/team-godspeed/include/framework/binder.h

4.2 godspeed::Debouncer Class Reference

A class for debouncing boolean data using a delay.

```
#include <debouncer.h>
```

Collaboration diagram for godspeed::Debouncer:



Public Member Functions

- Debouncer (int timeout=100)

  Constructor.
- bool Debounce ()

  Call whenever the signal switches on, and used the returned value as the actual debounced value.

Public Attributes

- int Timeout

  Delay used for debouncing.

4.2.1 Detailed Description

A class for debouncing boolean data using a delay.

The documentation for this class was generated from the following file:

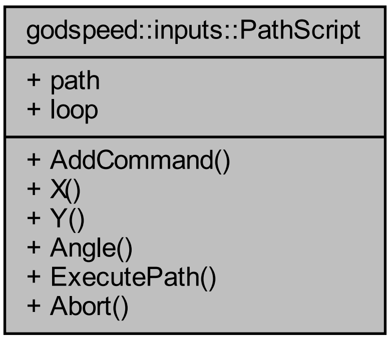
- C:/Users/anzel/source/repos/team-godspeed/include/framework/debouncer.h

4.3 godspeed::inputs::PathScript Class Reference

A class for creating scripted paths for the robot to take.

```
#include <path-script.h>
```

Collaboration diagram for godspeed::inputs::PathScript:



Public Member Functions

- void AddCommand (double x, double y, double a, double duration)

    Adds a command to the path script.

Static Public Member Functions

- static double **X** ()
- static double **Y** ()
- static double **Angle** ()
- static void ExecutePath (PathScript &path)

    Starts execution of the given path.
- static void Abort ()

    Flags the current path to stop execution once the current update cycle is finished.

Public Attributes

- std::list< COMMAND_TUPLE > path

  List of commands of form x-speed, y-speed, angle-speed, duration.
- bool loop

  Set this to true if you wish the path script to repeat after finishing.

4.3.1 Detailed Description

A class for creating scripted paths for the robot to take.

The paths are scripted using "commands" which are tuples consisting of an x-speed, a y-speed, an angular speed, and a duration. First create an instance of PathScript, then add all commands you want, then call PathScript::ExecutePath() passing in the PathScript you just created as the argument. This will immediately begin execution of the path. Path execution proceeds by recursive time-delayed updates which update variables, then whatever is bound the X(), Y(), and Angle() will have access to the updated variables. So the udpates are asyncronous from the actual movement changes and may not be deterministic.

The documentation for this class was generated from the following files:

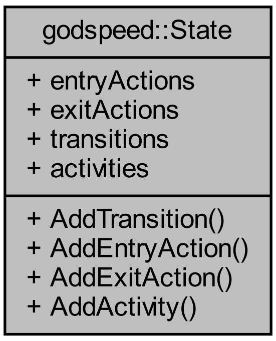
- C:/Users/anzel/source/repos/team-godspeed/include/inputs/path-script.h
- C:/Users/anzel/source/repos/team-godspeed/src/inputs/path-script.cpp

4.4 godspeed::State Class Reference

A state class used to create statemachines to execute using the StateMachine namespace.

```
#include <state-machine.h>
```

Collaboration diagram for godspeed::State:



Public Member Functions

- void AddTransition (bool(∗condition)(void), State &state)

  Add a transition to the state.
- void AddEntryAction (void(∗action)(void))

  Add an entry action to the state.
- void AddExitAction (void(∗action)(void))

  Add an exit action to the state.
- void AddActivity (Binding &activity)

  Add an activity (binding) to the state.

Public Attributes

- std::list< ACTION > **entryActions**
- std::list< ACTION > **exitActions**
- std::list< TRANSITION > **transitions**
- std::list< Binding ∗ > **activities**

4.4.1 Detailed Description

A state class used to create statemachines to execute using the StateMachine namespace.

The documentation for this class was generated from the following file:

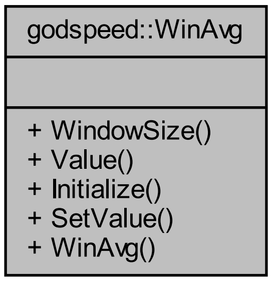
- C:/Users/anzel/source/repos/team-godspeed/include/autonomous/state-machine.h

4.5 godspeed::WinAvg Class Reference

A class to apply windowed averaging.

```
#include <smoothing.h>
```

Collaboration diagram for godspeed::WinAvg:



Public Member Functions

- int WindowSize ()

  Size of the averaging window.
- double Value ()

  Returns average of current window.
- void Initialize (double val)

  Set value of all points in window.
- void SetValue (double val)

  Add new data point to window.
- WinAvg (int winsize=100)

  Constructor.

4.5.1 Detailed Description

A class to apply windowed averaging.

The documentation for this class was generated from the following file:

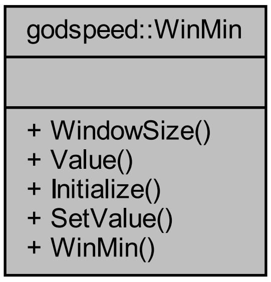
- C:/Users/anzel/source/repos/team-godspeed/include/framework/smoothing.h

4.6 godspeed::WinMin Class Reference

A class to apply windowed minimum.

```
#include <smoothing.h>
```

Collaboration diagram for godspeed::WinMin:



Public Member Functions

- int WindowSize ()

  Size of the window.
- double Value ()

  Returns minimum of current window.
- void Initialize (double val)

  Set value of all points in window.
- void SetValue (double val)

  Add new data point to window.
- WinMin (int winsize=100)

  Constructor.

4.6.1 Detailed Description

A class to apply windowed minimum.

The documentation for this class was generated from the following file:

- C:/Users/anzel/source/repos/team-godspeed/include/framework/smoothing.h

Index