Contents

1

1

2

2

3

#define **SKILLS_H**

4 class **AdvSkills**

all basic skills of the player

Public Members

all basic skills of the player. All skills calculate and execute one action leading to a target given by the agent. They return a state from which the agent can decide wether another action is needed or not.

4.1 variable constants.

Names

	double	**precision**	*assumed number precision. Default: 1e-5*	
4.1.1	double	**kickRange**	*the range in which a kick is likely successful*	7
4.1.2	double	**minSpeed**	*velocities below this are considered zero*	7
4.1.3	double	**prepareOuterLimit**	*how far player is placed from ball by prepare before going for it*	7
4.1.4	double	**prepareInnerLimit**	*distance player should keep to ball when preparing a kick in before directly intercepting the ball*	8
	int	**minInterceptTime**	*time until that intercept times should be calculated*	
	double	**goToThreshStretch**	*angle threshold for goToPosition. Lower values mean higher correction*	
4.1.5	double	**turnMinAngle**	*turn(Heads) below this are marked FINISHED*	8
	int	**turnLatency**	*time to wait between two turnHeads if last seems not to be executed*	
	double	**directThreshold**	*improvement threshold to choose direct kick over indirect*	
	double	**laybackErr**	*position error to actually lay back*	
4.1.6	double	**innerSafeCircle**	*the inner circle we do not want to get in*	8
	double	**safeCircle**	*the distance where the ball is handled optimally*	
	double	**stopableV**	*the velocity which the ball may get maximally so that it can be handled*	
4.1.7	double	**dribbleBodyErr**	*the tolerated body direction angle error in dribble*	8

these are values that normally should not change permanently during a game, but that may need to be adjusted to the specific settings of a match or of the agent using them.

4.1.1

double **kickRange**

the range in which a kick is likely successful

the range in which a kick is likely successful. Default: .87(PLAYER_SIZE+BALL_SIZE+KICKABLE_MARGIN)

4.1.2

double **minSpeed**

velocities below this are considered zero

velocities below this are considered zero. Default: 0.1

4.1.3

double **prepareOuterLimit**

how far player is placed from ball by prepare before going for it

how far player is placed from ball by prepare before going for it. Default: 3*kickRange

4.1.4

double **prepareInnerLimit**

distance player should keep to ball when preparing a kick in before directly intercepting the ball

distance player should keep to ball when preparing a kick in before directly intercepting the ball. Default: 2*collRange

4.1.5

double **turnMinAngle**

turn(Heads) below this are marked FINISHED

turn(Heads) below this are marked FINISHED. Default: 15

4.1.6

double **innerSafeCircle**

the inner circle we do not want to get in

the inner circle we do not want to get in. Default: PLAYER_SIZE+BALL_SIZE + 0.3*KICKABLE_MARGIN

4.1.7

double **dribbleBodyErr**

the tolerated body direction angle error in dribble

the tolerated body direction angle error in dribble. Default: 10

4.1.8

double **dribbleHoldErr**

the tolerated ball direction angle error in dribble

the tolerated ball direction angle error in dribble. Default: 30

4.2

AdvSkills (Comm *comm)

the constructor

the constructor.

Parameters: `comm` — used for sending the calculated commands.

4.3

kicking commands

Names

KickMode	*compatability - translate modes into time lengths*

4.3.1 **KickState**

KRANGE *ball out of kick range*

KFINISHING *ball will have wanted velocity after this kick*

KTOOSHORT *ball will be slower than wanted*

KPLACING *ball is placed to allow a better kick*

KPLACEFORSKICK *ball is layed back*

KSTOPPING *requested velocity is near 0, so stopping ball*

. . . . 11

4.3.2 **KickCommand** 12

bool **kickSafe** (const Snapshot *snap)
returns if a kick command is likely to be successful

4.3.3 ExtKickCommand
kick (const Snapshot *current_snap,
const Vector2d &target_v, int max_time = 2)
try to give the ball a target velocity
. . . . 12

4.3.4 ExtKickCommand
kick (const Snapshot *current_snap,
double target_v, const Vector2d &target_pos,
int max_time = 2)
kick to an absolute target position
. . . . 13

4.3.5 ExtKickCommand
rekick (const Snapshot *current_snap,
const ExtKickCommand &kcm)
does the last kick command again

13

4.3.6 KickCommand
placeBall (const Snapshot *current_snap,
const Vector2d &target)
try to place ball on a safe circle around the player 14

4.3.7 KickCommand

4.3.1

KickState

KRANGE	*ball out of kick range*
KFINISHING	*ball will have wanted velocity after this kick*
KTOOSHORT	*ball will be slower than wanted*
KPLACING	*ball is placed to allow a better kick*
KPLACEFORSKICK	*ball is layed back*
KSTOPPING	*requested velocity is near 0, so stopping ball*

KRANGE	ball out of kick range
KFINISHING	ball will have wanted velocity after this kick
KTOOSHORT	ball will be slower than wanted
KPLACING	ball is placed to allow a better kick
KPLACEFORSKICK	ball is layed back
KSTOPPING	requested velocity is near 0, so stopping ball

4.3.2

KickCommand

Members

Vector2d	**v**	*calculated ball velocity*
KickState	**state**	

4.3.3

ExtKickCommand **kick** (const Snapshot *current_snap, const Vector2d &target_v, int max_time = 2)

try to give the ball a target velocity

Arguments

int	**time**	*how many steps it will likely cost to complete the kick*
int	**max_time**	
Vector2d	**target**	
double	**target_v**	*if >= 0, the target velocity, indicates position kick*

try to give the ball a target velocity. Collisions are avoided by kicking the ball around the player if necessary. It is possible to let kick try to place the ball behind the player for a really hard kick (superkick).

It is possible, that you have to call kick more than once. So call *rekick* to continue the last kick until it says RANGE, so that as long as possible kicks are sent.

Parameters:

`current_snap` — the snapshot the agent believes to be current
`target_v` — the velocity the ball should get
`max_time` — maximal time until kick must be finished

4.3.4

ExtKickCommand **kick** (const Snapshot *current_snap, double target_v, const Vector2d &target_pos, int max_time = 2)

kick to an absolute target position

kick to an absolute target position.

Parameters:
`current_snap` — the snapshot the agent believes to be current
`target_pos` — the absolute coordinates where the ball should be kicked.
`target_v` — the velocity the ball should have when reaching the target position

4.3.5

ExtKickCommand **rekick** (const Snapshot *current_snap, const ExtKickCommand &kcm)

does the last kick command again

does the last kick command again.

Parameters:
`current_snap` — the snapshot the agent believes to be current
`kcm` — the KickCommand from kick to continue

4.3.6

KickCommand **placeBall** (const Snapshot *current_snap, const Vector2d &target)

try to place ball on a safe circle around the player

try to place ball on a safe circle around the player. The balls velocity will be slow enough to be stopped within one turn, if possible.

Parameters: `current_snap` — the snapshot the agent believes to be current
`target` — the wanted position of the ball relative to the player in absolute coordinates.

4.3.7

KickCommand **stopBall** (const Snapshot *current_snap)

try to stop ball

try to stop ball.

Parameters: `current_snap` — the snapshot the agent believes to be current

4.4

intercepting skills

Names

4.4.1 **TurtleState**

TFINISHED *at target position/ball*

TONWAY *player is on way, but not in target position*

TGOINGBACKWARDS *player shows in opposite direction*

. . . . 15

4.4.1 TurtleState

TFINISHED *at target position/ball*

TONWAY *player is on way, but not in target position*

TGOINGBACKWARDS *player shows in opposite direction*

TFINISHED	at target position/ball
TONWAY	player is on way, but not in target position
TGOINGBACKWARDS	player shows in opposite direction

4.4.2

TurtleCommand

Members

int	**time**	*estimated time until finished*
Vector2d	**v**	*calculated player velocity*
Angle	**bodyDir**	*players new body dir*
double	**turnMoment**	*the angle to turn to reach bodyDir (affected by inertia moment)*
TurtleState	**state**	

4.4.3

TurtleCommand **controlBall** (const Snapshot *current_snap, double stamina, double error, WorldBall &meet)

try to get the ball under control, that is, intercept it

try to get the ball under control, that is, intercept it.

Parameters:
`current_snap` — the snapshot the agent believes to be current
`stamina` — the stamina to use at most this step
`meet` — the position we will meet the ball

4.4.4

TurtleCommand **goToPosition** (const Snapshot *current_snap, const Vector2d &target, double error = 5.0, double stamina = STAMINA_INC_MAX, bool dashBackwardsIfNeeded = false, Angle prefAngle = 0)

go to a position

go to a position. This is intended for long range intercepts and absolute positions, so error is assumed to be relative large (> PLAYER_SIZE). TIME INFORMATION OF TURTLECOMMAND IS NOT SET!!!

Parameters:

`current_snap` — the snapshot the agent believes to be current
`target` — absolute target position on field
`error` — radius around the target in which the player should end up
`stamina` — the stamina to use at most this step
`dashBackwardsIfNeeded` — keep the difference between bodyDir and prefAngle < 90 by eventually dashing backwards
`prefAngle` — the angle to keep if minAngle is set

4.4.5

TurtleCommand **goToPositionTimed** (const Snapshot *current_snap, const Vector2d &target, double stamina, double error)

go to a position

go to a position.

Parameters:

`current_snap` — the snapshot the agent believes to be current
`target` — absolute target position on field
`stamina` — the stamina to use for the complete intercept additional to the stamina gained during the intercept.
`error` — radius around the target in which the player should end up

4.5

dribbling

Names

4.5.1 **DribbleState**

DOK *simply dribbling*

DGOINGFORBALL *ball lost,*
actually intercepting it

.... 19

4.5.2 **DribbleCommand** 19

4.5.3 DribbleCommand
dribble (const Snapshot *current_snap,
Angle dribbledir, Angle hold_dir,
double stamina = STAMINA_INC_MAX)
dribble into an absolute direction 20

void **exec** (const Snapshot *current_snap,
const DribbleCommand &)

4.5.1

DribbleState

DOK	*simply dribbling*
DGOINGFORBALL	*ball lost, actually intercepting it*

DOK	simply dribbling
DGOINGFORBALL	ball lost, actually intercepting it

4.5.2

DribbleCommand

Members

Vector2d	**v_p**	*calculated player and ball velocity*
Angle	**bodyDir**	
DribbleState	**state**	

4.5.3

DribbleCommand **dribble** (const Snapshot *current_snap, Angle dribbledir, Angle hold_dir, double stamina = STAMINA_INC_MAX)

dribble into an absolute direction

dribble into an absolute direction. The agent can decide in which direction to the player the ball should be kept (in absolute direction).

Parameters:
`current_snap` — the snapshot the agent believes to be current
`dribbledir` — the direction in which to dribble
`hold_dir` — the direction we want to hold the ball from us

4.6

goalie's catch skill

Names

command storage

4.6.1 **CatchState**

CBAN *catch still banned*

CRANGE *ball to far away or outside penalty area*

CNOTGOALIE *player is not goalie and may not catch*

COK *done*

. . . . 21

bool **catchSafe** (const Snapshot *current_snap)
returns if a catch command is likely to be successful

4.6.2 **CatchCommand** 21

4.6.3 CatchCommand **goalieCatch** (const Snapshot *current_snap)
catch the ball, if allowed 22

void **exec** (const Snapshot *current_snap,
const CatchCommand &)

4.6.1

CatchState

CBAN *catch still banned*

CRANGE *ball to far away or outside penalty area*

CNOTGOALIE *player is not goalie and may not catch*

COK *done*

CBAN catch still banned

CRANGE ball to far away or outside penalty area

CNOTGOALIE player is not goalie and may not catch

COK done

4.6.2

CatchCommand

Members

Angle **catchDir**

CatchState **state**

4.6.3

CatchCommand **goalieCatch** (const Snapshot *current_snap)

catch the ball, if allowed

catch the ball, if allowed. This is checked via worldmodel.

Parameters: `current_snap` — the snapshot the agent believes to be current

4.7

turn

Names

4.7.1 TurnState

ROK *done*

RANGLE *angle is too large, truncated*

RFINISHED *angle to turn is small*

ROK done

RANGLE angle is too large, truncated

RFINISHED angle to turn is small

4.7.2 TurnCommand

Members

Angle	**bodyDir**	*the optimal body dir actually achievable*
double	**turnMoment**	*the angle to turn to reach bodyDir (affected by inertia moment)*
TurnState	**state**	

4.7.3

TurnCommand **turn** (const Snapshot *current_snap, Angle dir)

turn into absolute direction dir

turn into absolute direction dir.

Parameters: `current_snap` — the snapshot the agent believes to be current

4.8

turn neck

Names

turnHeadToBall (const Snapshot *current_snap, const TurnCommand &tcm)
turn to ball considering a TurnCommand 26

4.8.5 TurnHeadCommand
turnHeadToBall (const Snapshot *current_snap, const TurtleCommand &tcm)
turn to ball considering a TurtleCommand 27

4.8.6 TurnHeadCommand
turnHeadForDribble (const Snapshot *current_snap, const DribbleCommand &dcm, double dribbledir)
turn to ball considering a DribbleCommand 27

void **exec** (const Snapshot *current_snap, const TurnHeadCommand &)

4.8.1

TurnHeadState

THOK *done*

THANGLE *angle is too large, truncated*

THBAN *last turn_neck seems not to be committed, waiting one turn*

THFINISHED *angle to turn small*

THOK done

THANGLE angle is too large, truncated

THBAN last turn_neck seems not to be committed, waiting one turn

THFINISHED angle to turn small

4.8.2

TurnHeadCommand

Members

Angle	**faceDir**	*the optimal face dir actually achievable*
TurnHeadState	**state**	

4.8.3

TurnHeadCommand **turnHead** (const Snapshot *current_snap, Angle dir)

turns head into specified absolute direction

turns head into specified absolute direction. The resulting head angle is rounded to integer multiples of angleStepping to avoid too many turnHeads.

Parameters: `current_snap` — the snapshot the agent believes to be current
`dir` — the absolute dir we want to see

4.8.4

TurnHeadCommand **turnHeadToBall** (const Snapshot *current_snap, const TurnCommand &tcm)

turn to ball considering a TurnCommand

turn to ball considering a TurnCommand

4.8.5

TurnHeadCommand **turnHeadToBall** (const Snapshot *current_snap, const TurtleCommand &tcm)

turn to ball considering a TurtleCommand

turn to ball considering a TurtleCommand

4.8.6

TurnHeadCommand **turnHeadForDribble** (const Snapshot *current_snap, const DribbleCommand &dcm, double dribbledir)

turn to ball considering a DribbleCommand

turn to ball considering a DribbleCommand

Parameters: `dribbledir` — should be the same as for the dribble command

4.9

void **changeView** (const Snapshot *current_snap, view_width width)

changes view only when needed

changes view only when needed.

Parameters: `current_snap` — the snapshot the agent believes to be current
`width` — the wanted view width

4.10

void **changeView** (const Snapshot *current_snap, view_quality quality)

changes view only when needed

changes view only when needed.

Parameters: `current_snap` — the snapshot the agent believes to be current
`quality` — the wanted view quality

4.11

void **changeView** (const Snapshot *current_snap, view_width width, view_quality quality)

changes view only when needed

changes view only when needed.

Parameters: `current_snap` — the snapshot the agent believes to be current
`width` — the wanted view width
`quality` — the wanted view quality

4.12

low-level commands.

Names

void	**noAction** ()
void	**basicDash** (double pow)
void	**basicKick** (double pow, double dir)
void	**basicTurn** (double pow)
void	**basicCatch** (double dir)
void	**basicMove** (double x, double y)
void	**sendTurnNeck** (double dir)
void	**sendChangeView** (view_width width, view_quality quality)
void	**sendSenseBody** ()

This are all commands that are directly aviable from comm.C

Class Graph