# Basic Command for A Client

Ke Shi

Multi-Agent Systems Lab.
Department of Computer Science and Technology
University of Science and Technology of China


Oct. 25, 2008

- 该命令使球员改变身体方向。

- 该命令使球员改变身体方向。
- @param double moment

- 该命令使球员改变身体方向。
- @param double moment
- actual_angle =
  moment / (1.0 + inertia_moment * player_speed);

- 该命令使球员改变身体方向。
- @param double moment
- actual_angle =
  moment / (1.0 + inertia_moment * player_speed);
- 球员的inertia_moment的默认值是5。

- 该命令使球员向前或向后奔跑。

- 该命令使球员向前或向后奔跑。
- @param double power

- 该命令使球员向前或向后奔跑。
- @param double power
- 该命令给球员一个加速度，加速度方向和球员的身体朝向相同或相反。另外，每个球员都有一定的体力，会因为dash命令而消耗。

- 该命令使球员向前或向后奔跑。
- @param double power
- 该命令给球员一个加速度，加速度方向和球员的身体朝向相同或相反。另外，每个球员都有一定的体力，会因为dash命令而消耗。
- 在上下半场开始时，球员体力被置为stamina_max。如果球员向前加速（power $>$ 0），体力值降低power；如果是向后加速，体力值降低两倍的power。

- 体力降低后，SoccerServer会计算dash命令中power的有效部分，命令dash的有效部分（eff_dash_power）是由dash_power_rate和球员当前的effort决定的。

- 体力降低后，SoccerServer会计算dash命令中power的有效部分，命令dash的有效部分（eff_dash_power）是由dash_power_rate和球员当前的effort决定的。
- eff_dash_power =
  effort * power * dash_power_rate;

- 体力降低后，SoccerServer会计算dash命令中power的有效部分，命令dash的有效部分（eff_dash_power）是由dash_power_rate和球员当前的effort决定的。
- eff_dash_power =
  effort * power * dash_power_rate;
- 算出来的eff_dash_power和球员的身体方向一起转化为矢量作为球员本周期的加速度。

# 体力模型

- 体力模型中有3个重要的值：体力值stamina，恢复recovery，效力effort。

# 体力模型

- 体力模型中有3个重要的值：体力值stamina，恢复recovery，效力effort。
- 执行dash时会降低体力值，在每个周期体力值会有少量的增加。

# 体力模型

- 体力模型中有3个重要的值：体力值stamina，恢复recovery，效力effort。
- 执行dash时会降低体力值，在每个周期体力值会有少量的增加。
- recovery表明每个周期可以恢复多少体力，effort表明执行dash时的效力问题。

# 体力模型

- 体力模型中有3个重要的值：体力值stamina，恢复recovery，效力effort。
- 执行dash时会降低体力值，在每个周期体力值会有少量的增加。
- recovery表明每个周期可以恢复多少体力，effort表明执行dash时的效力问题。
- 如果在比赛的某个周期，体力值stamina低于某个极限，那么effort和recovery将会不断减少直至最少值为止。如果球员体力是某个极限之上，effort将会增大直到最大值，但是recovery将不会再增加。

- 该命令使球员踢球。

- 该命令使球员踢球。
- @param double power
  @param double dir

- 该命令使球员踢球。
- @param double power
  @param double dir
- 该命令有两个参数，分别是踢球力量和踢球角度。

- 该命令使球员踢球。
- @param double power
  @param double dir
- 该命令有两个参数，分别是踢球力量和踢球角度。
- 只要球的中心位置到球员的中心位置小于（player_size + kickable_margin），该命令就会被执行。

- eff_power =
  power *
  kick_power_rate *
  (1.0 - 0.25 * dir_diff / M_PI - 0.25 * dist_ball /
  kickable_margin);

- eff_power =
  power *
  kick_power_rate *
  (1.0 - 0.25 * dir_diff / M_PI - 0.25 * dist_ball /
  kickable_margin);
- accel =
  Polar2Vector(eff_power, dir + body_angle);

# kick的误差

- pos_rate =
  0.5 +
  0.25 * ( dir_diff / M_PI + dist_ball / kickable_margin);

# kick的误差

- pos_rate =
  0.5 +
  0.25 * ( dir_diff / M_PI + dist_ball / kickable_margin);
- speed_rate =
  0.5 +
  0.5 * (ball_speed / (ball_speed_max * ball_decay));

# kick的误差

- pos_rate =
  0.5 +
  0.25 * ( dir_diff / M_PI + dist_ball / kickable_margin);
- speed_rate =
  0.5 +
  0.5 * (ball_speed / (ball_speed_max * ball_decay));
- max_rand =
  kick_rand *
  (power / max_power) *
  (pos_rate + speed_rate);

- 该命令使球员铲球。

- 该命令使球员铲球。
- @param double power_or_angle

- 该命令使球员铲球。
- @param double power_or_angle
- 如果是以version小于12版本以前的client连接SoccerServer，铲球的参数为power；如果是以version等于12版本的client连接SoccerServer，铲球的参数为angle。这里只介绍后者的模型。

# 铲球概率

- tackle_dist =
  (ball_2_player.x $>$ 0.0) ?
  ServerParam::instance().tackleDist() :
  ServerParam::instance().tackleBackDist();

# 铲球概率

- tackle_dist =
  (ball_2_player.x > 0.0) ?
  ServerParam::instance().tackleDist() :
  ServerParam::instance().tackleBackDist();
- tackle_fail_prob =
  pow(fabs(ball_2_player.x) / tackle_dist, tackle_exponent) +
  pow(fabs(ball_2_player.y) / tackle_width, tackle_exponent);

- eff_power =
  (max_back_tackle_power +
  (max_tackle_power - max_back_tackle_power) *
  (1.0 - (fabs(angle) / M_PI))) *
  tackle_power_rate;

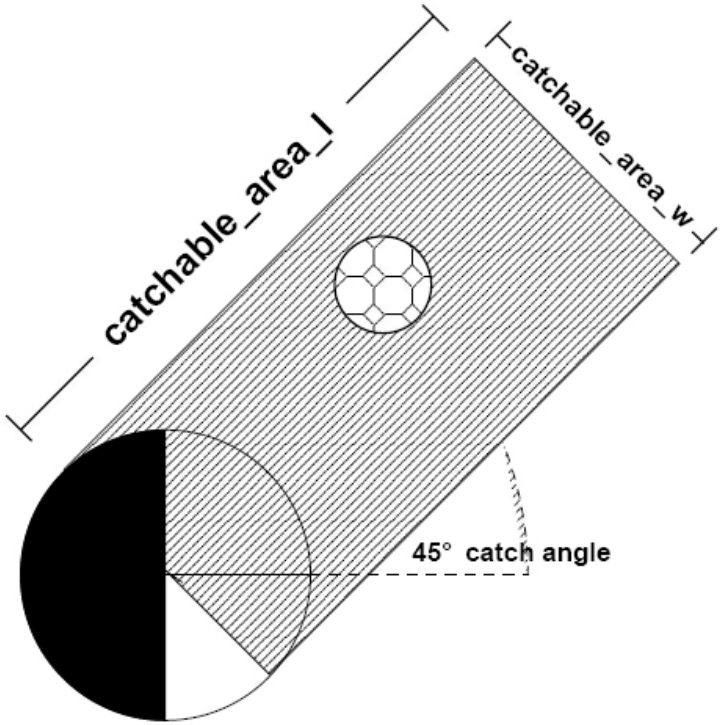
- eff_power =
  (max_back_tackle_power +
  (max_tackle_power - max_back_tackle_power) *
  (1.0 - (fabs(angle) / M_PI))) *
  tackle_power_rate;
- eff_power *=
  1.0 - 0.5 * (fabs(ball_2_player.Dir()) / M_PI);

- eff_power =
  (max_back_tackle_power +
  (max_tackle_power - max_back_tackle_power) *
  (1.0 - (fabs(angle) / M_PI))) *
  tackle_power_rate;
- eff_power *=
  1.0 - 0.5 * (fabs(ball_2_player.Dir()) / M_PI);
- accel =
  Polar2Vector(eff_power, angle + body_angle);

- 该命令使守门员扑球。

- 该命令使守门员扑球。
- @param double dir

- 该命令使守门员扑球。
- @param double dir
- 守门员是唯一能执行catch命令的球员。守门员可以从任何方向扑到球，只要球在可扑范围内，守门员在禁区内而且比赛模式是play_on。


catchable_area_l
catchable_area_w
45° catch angle

- TurnNeck

- TurnNeck
- Say(char *msg)

- TurnNeck
- Say(char *msg)
- Attentionto(char side, int unum)

- TurnNeck
- Say(char *msg)
- Attentionto(char side, int unum)
- Pointto(double dist, double angle)

- TurnNeck
- Say(char *msg)
- Attentionto(char side, int unum)
- Pointto(double dist, double angle)
- Move(double x, double y)

- TurnNeck
- Say(char *msg)
- Attentionto(char side, int unum)
- Pointto(double dist, double angle)
- Move(double x, double y)
- ChangeView("narrow"|"normal"|"wide")

- TurnNeck
- Say(char *msg)
- Attentionto(char side, int unum)
- Pointto(double dist, double angle)
- Move(double x, double y)
- ChangeView("narrow"|"normal"|"wide")
- Compression(int level)

- SenseBody()

- SenseBody()
- Score()

- SenseBody()
- Score()
- Bye()

- SenseBody()
- Score()
- Bye()
- Done()

- SenseBody()
- Score()
- Bye()
- Done()
- Clang(int min_ver, int max_ver)

- SenseBody()
- Score()
- Bye()
- Done()
- Clang(int min_ver, int max_ver)
- Ear(char side, "on"|"off", "our"|"opp", "partial"|"complete")

- SenseBody()
- Score()
- Bye()
- Done()
- Clang(int min_ver, int max_ver)
- Ear(char side, "on"|"off", "our"|"opp", "partial"|"complete")
- SynchSee()

- SenseBody()
- Score()
- Bye()
- Done()
- Clang(int min_ver, int max_ver)
- Ear(char side, "on"|"off", "our"|"opp", "partial"|"complete")
- SynchSee()
- ChangePlayerType(int unum, int type_id)

# Thank you for your attention!
# Q & A