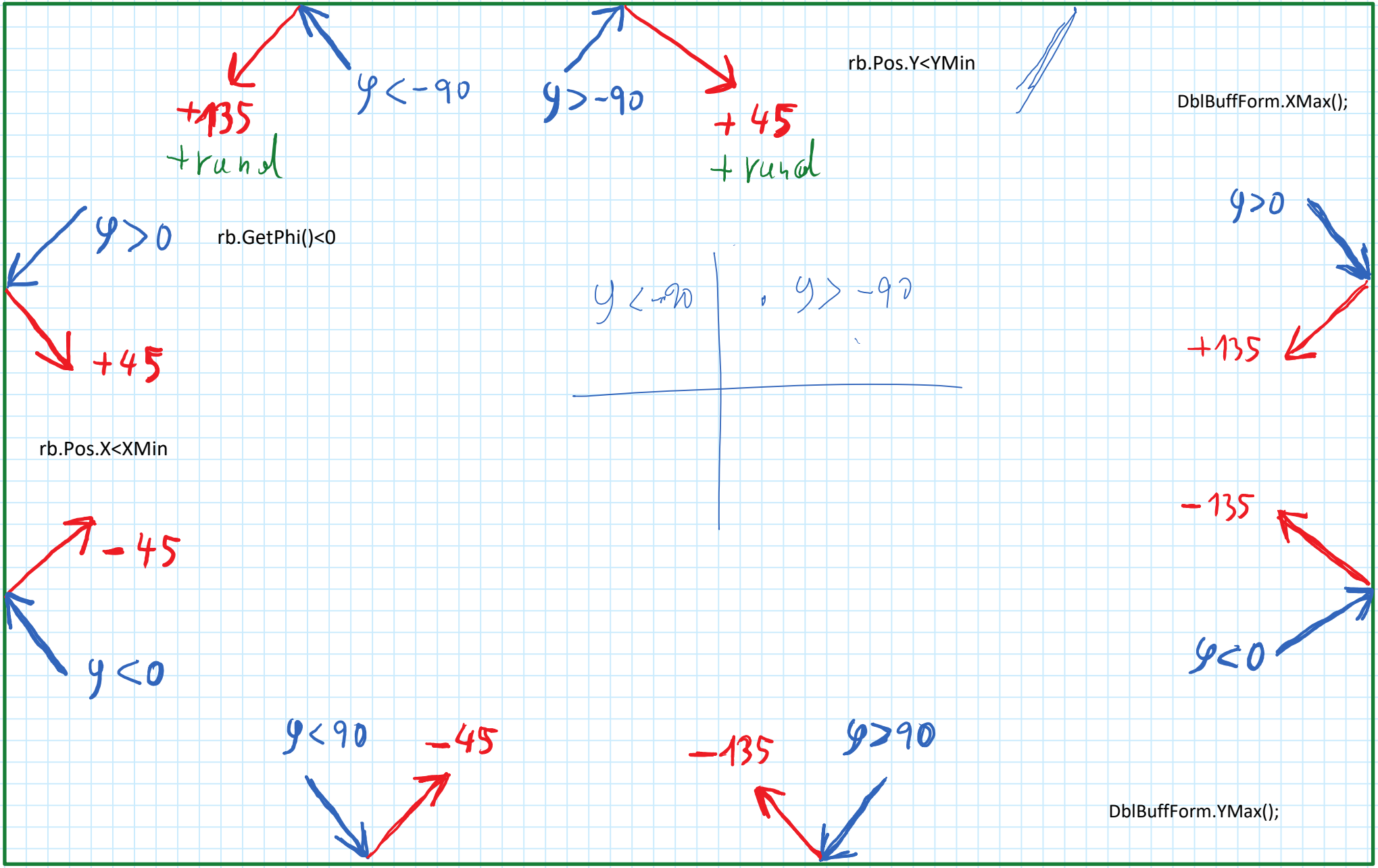
Aufgaben

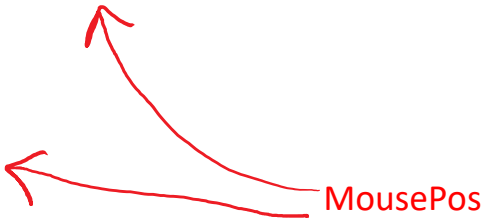
1. MausZug mit Minimal-Abstand
2. FollowPoints
3. FollowPoints mit Minimal-Abstand

Reflect at Border

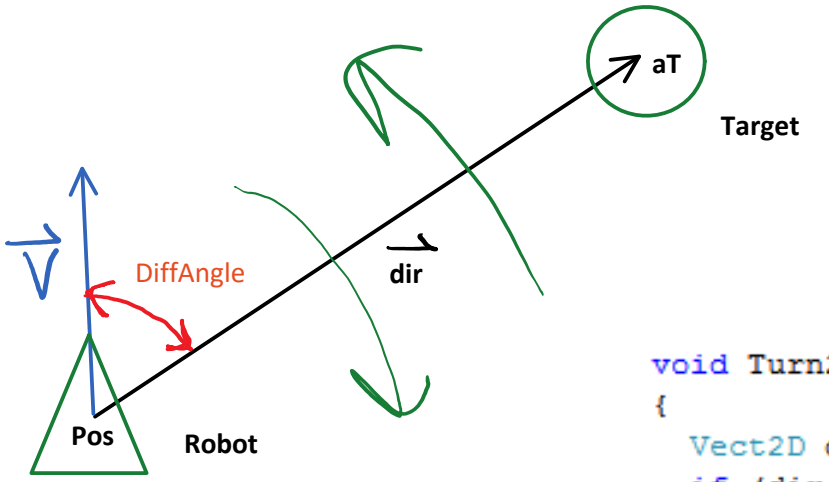


6a Reflect at Border some Tools and Tips

```
void PrgReflectBorder()
{ // Rasenmäher
  const double F_POW = 8; // Foreward
  const double T_POW = 5; // Turn
  const double R_POW = -2; // Reverse

  // rb.Pos.X <= 0;
  // rb.Pos.Y <= 0;

  DblBuffForm.XMax();
  DblBuffForm.YMax();

  Vect2D tmp = mpos;
}

// oder
RobotProg.mpos
```



7 Turn2Target und DiffAngle

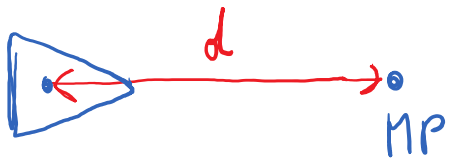



```
void Turn2Target(Vect2D aT, double aRotSpeed)
{
  Vect2D dir = Vect2D.Create(rb.Pos, aT);
  if (dir.DiffAngle(rb.V) < 0)
  {
  }
  else // dir.DiffAngle(rb.V) > 0
  {
  }
}
```

8 ChangePlace

```
void PrgChangePlace()
{
  // wait until sim is started
  WaitForUpdate(); WaitForUpdate();

  int i1 = Omgr.GetNearestIdx(rb.Pos);
  int i2;
  if (i1 == 0) i2 = 1; else i2 = 0;

  while (true)
  {
    Turn2Target(Omgr.At(i1).pos, 5);
    Thread.Sleep(500);
    // TurnRelAngle(40, 5);
    Drive2Target(Omgr.At(i1).pos, 6, 4);
    Thread.Sleep(500);

    Turn2Target(Omgr.At(i2).pos, 5);
    Thread.Sleep(500);
    // TurnRelAngle(40, 5);
    Drive2Target(Omgr.At(i2).pos, 6, 4);
    Thread.Sleep(500);
  }
}
```

9a Maus nachfahren mit Regelungstechnik

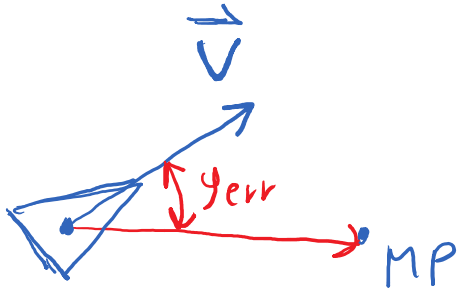



```
d = mpos.DistBetweenPoints(rb.Pos);
KP = 0.1;
rb.SetV( d*KP );
```

Die Motorleistung (Gesschw.) des Roboters wird
abhängig vom Abstand zur Mausposition gesetzt.
Je größer die Distanz zur Mausposition desto schneller
fährt der Roboter auf das Ziel zu.

KP ist ein Verstärkungsfaktor (Tuningfaktor) der bestimmt wie
stark der Abstand **d** auf die Motorleistung wirkt.

KP kann nicht beliebig groß gemacht werden
da die Motorleistung begrenzt ist.

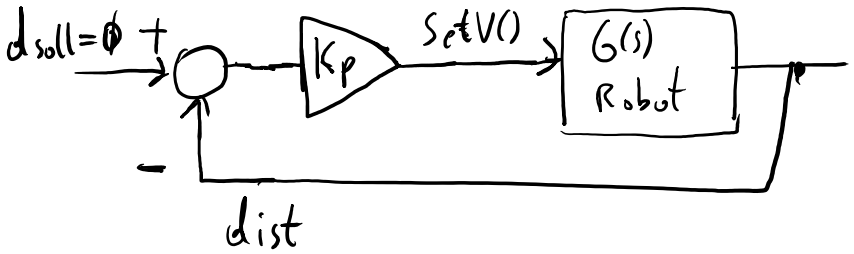


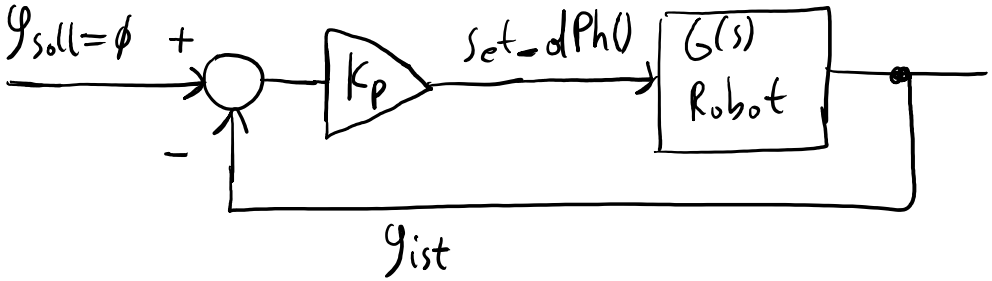
phiErr ist der Winkel zw. der Fahrrichtung des Robotes
und dem Richtungsvektor zur Mausposition.

Mit **phiErr** wird über einen P-Regler die Winkelgeschwindigkeit **dPhi**
des Roboters gesetzt.

```
KP = 0.2;
dir = mpos - rb.Pos;
phiErr = dir.DiffAngle(rb.V);
rb.Set_dPhi(phiErr  * KP);
```

10 FollowMouse Regelungstechnisch





12 Robot Train


```
void PrgTrain()
{
  // warten bis die Simulation gestartet wurde
  WaitForUpdate();
  // Methode im RbMgr um den Vordermann zu finden
  rbFront = RbMgr.FindNearestInFront(this);

  while(true)
  {
    WaitForUpdate();
    // Die gleiche Regelungstechnik wie bei FollowMouse
    CalcSpeed(rbFront);
  }
}
```