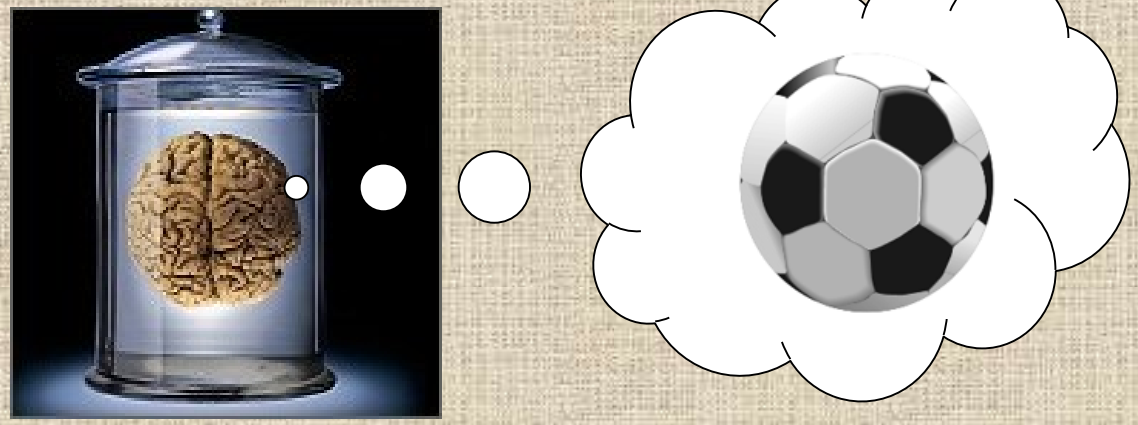


Soccer-Fun

functional programming with football

http://www.cs.ru.nl/P.Achten/SoccerFun/SoccerFun.html

What is Soccer-Fun?

- Program the brain of a footballer
- Build a team of eleven football players
- Play against other teams and see if your team is best
- Inspired by a citation from Johan Cruijff

Football brain

- Johan Cruijff (De Tijd, may 7 1982):

  "If I am in possession of the ball and I want to play it to another team member, I need to take my direct opponent into account, as well as the wind, the grass, and the speed with which other players run. We compute the strength with which to kick and the direction in which to do so in a tenth of a second. The computer needs two minutes for this!"

Voetbalbrein

- Johan Cruijff (De Tijd, 7 mei 1982):

  *"If I am in possession of the ball and I want to play it to another team member, I need to take my **direct opponent** into account, as well as the **wind**, the **grass**, and the **speed with which other players run**. We compute the **strength** with which to kick and the **direction** in which to do so in a tenth of a second. The computer needs two minutes for this!"*

- This computation is a function:

  brain :: ***Opponent Wind Grass Players*** → ***(Strength, Direction)***

Soccer-Fun

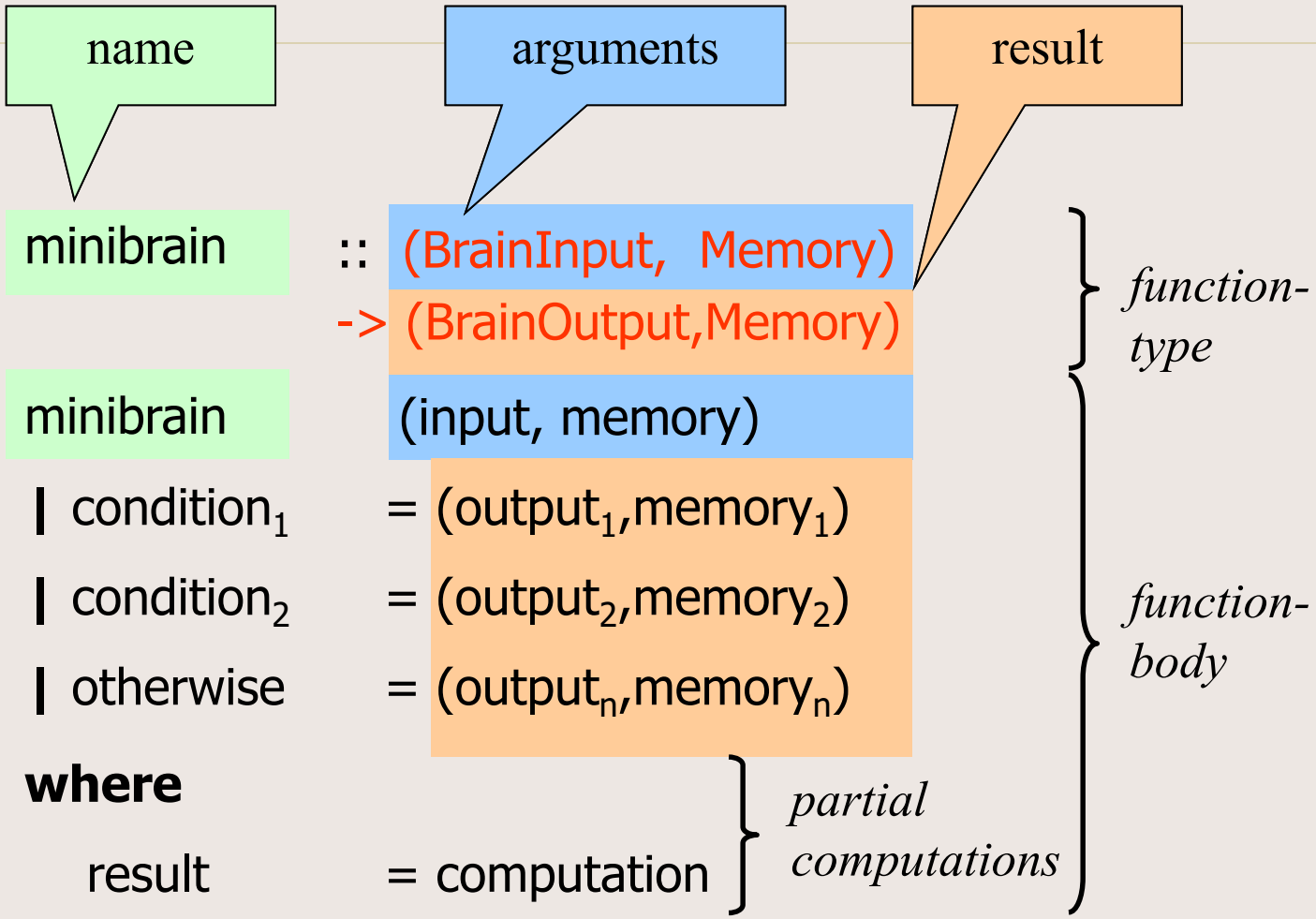

SoccerFun: Van Gaal's Electronic Notebook
File Game
MiniEffiesW 0 0 MiniEffiesE
Rounds/sec: 0
MiniF.5
MiniF.6
MiniF.8
MiniF.9
MiniF.2
MiniF.3
MiniF.10
MiniF.11
MiniF.1
MiniF.4
MiniF.7
MiniF.7
MiniF.4
MiniF.11
MiniF.10
MiniF.3
MiniF.2
MiniF.9
MiniF.8
MiniF.6
MiniF.5
1st half
0:40 min
Ivanov

Model football(er)s

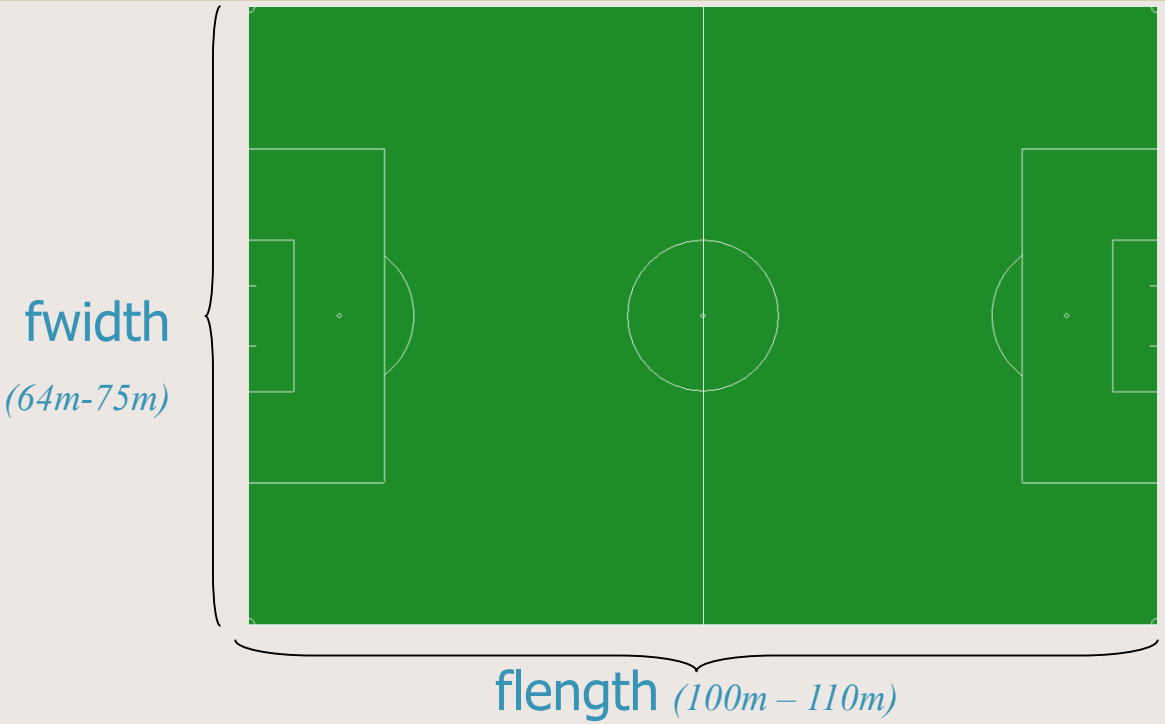
Footballer

```
:: Footballer = E.m:
        { playerID    :: FootballerID, nose :: Angle
        , pos         :: Position,     speed  :: Speed
        , name        :: String,       length :: Length
        , stamina     :: Stamina,      health :: Health
        , skills      :: MajorSkills
        , effect      :: Maybe FootballerEffect
        , brain       :: Brain (FootballerAI m) m
        }
```

“Our” brain is also a function



Dimensions – football field

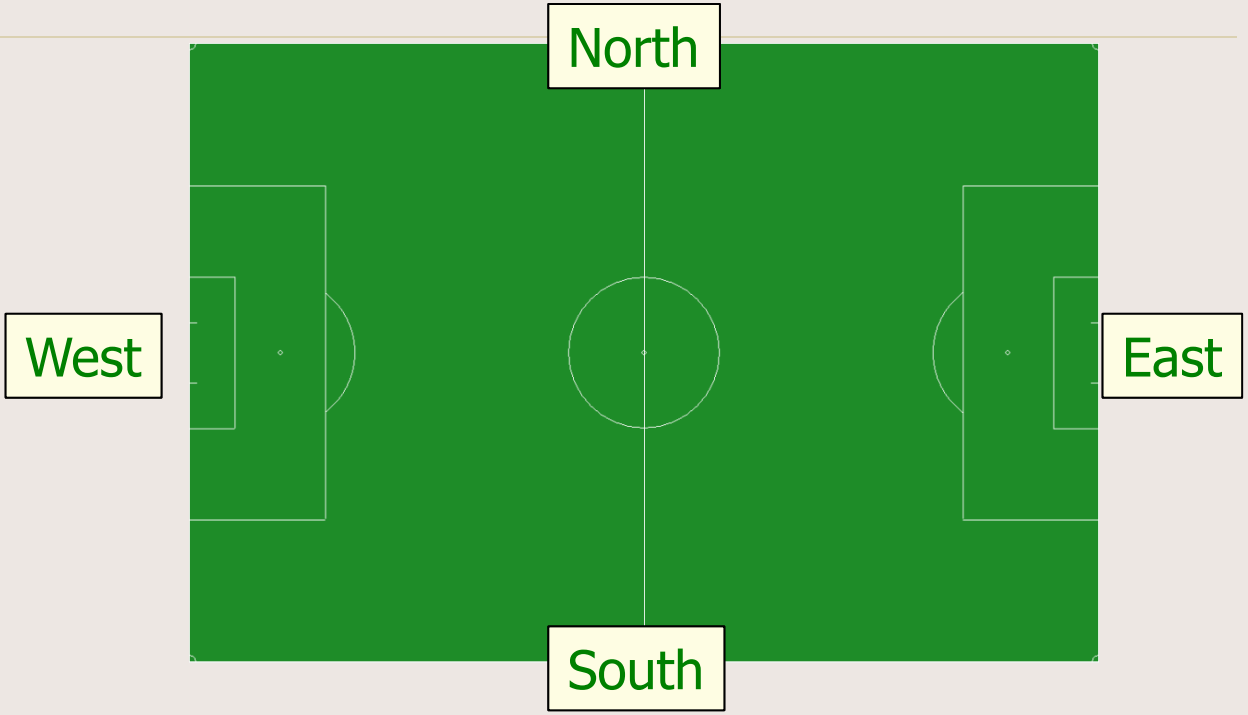



```
:: FootballField  = { fwidth :: FieldWidth, flength :: FieldLength}
:: FieldWidth     :== Metre
:: FieldLength    :== Metre
m :: Real -> Metre
```

Dimensions

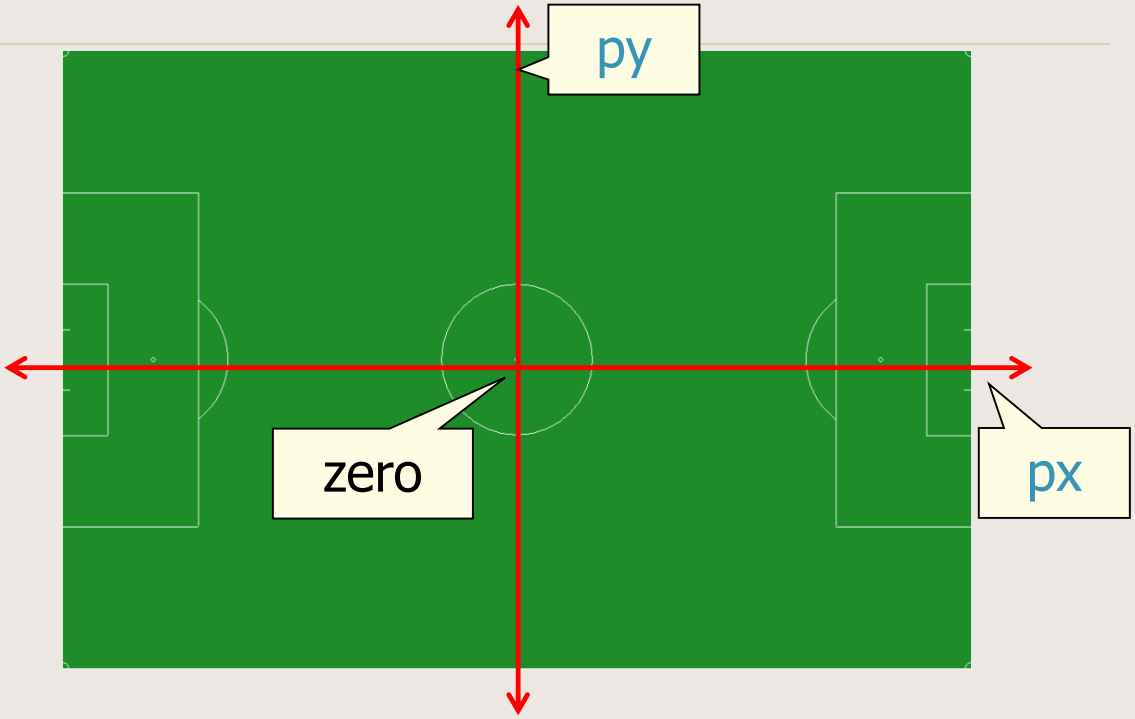
Eenheid	English	Soccer-Fun		Type
length	0 m 50.4 m	(m 0.0) (m 50.4)	or: zero	Metre
time	0 min 45 min	(minutes 0.0) (minutes 45.0)	or: zero	Minutes
angle	0° 90° 180° 270°	(degree 0) (degree 90) (degree 180) (degree 270)	or: zero or: (rad (0.5*pi)) or: (rad pi) or: (rad (1.5*pi))	Angle
velocity	0 m/s 10 m/s	(ms 0.0) (ms 10.0)	or: zero	Velocity

Dimensions – football field



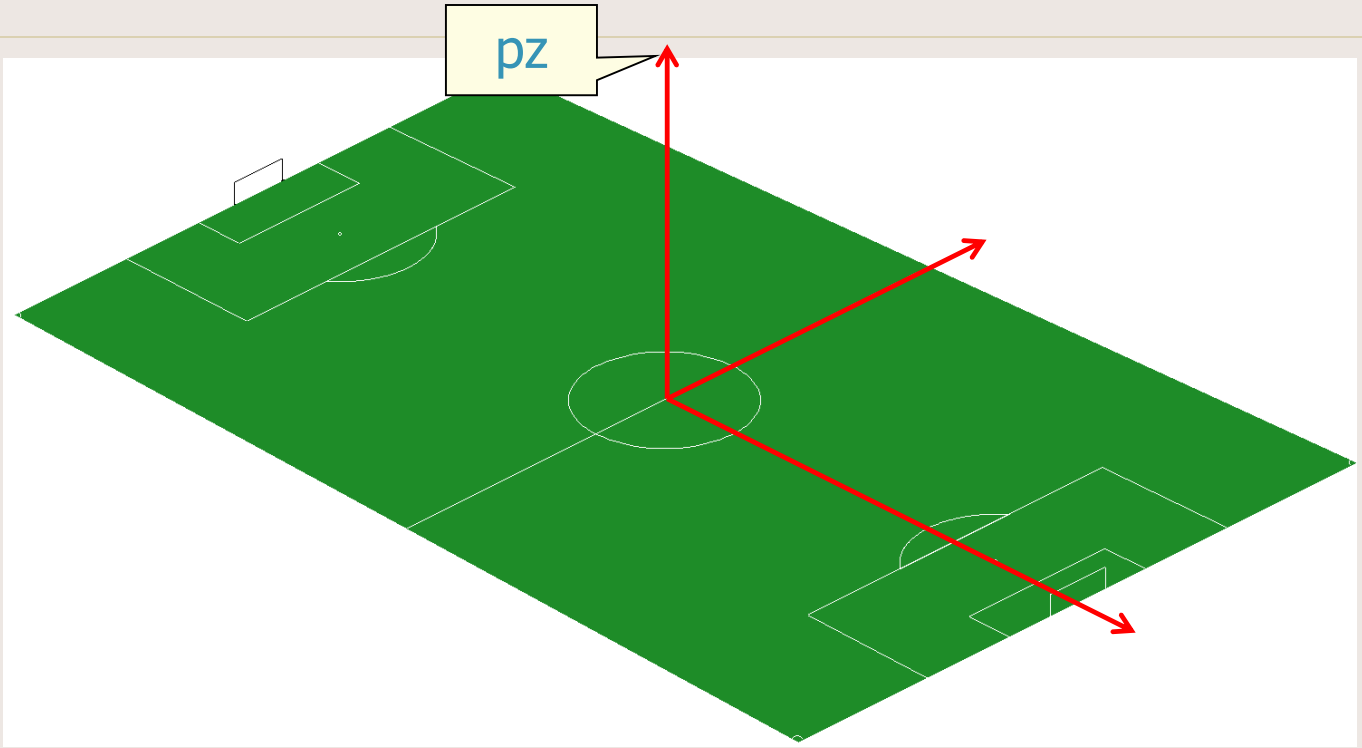
:: Edge = North | South
:: Home= West | East

Dimensions – coordinates



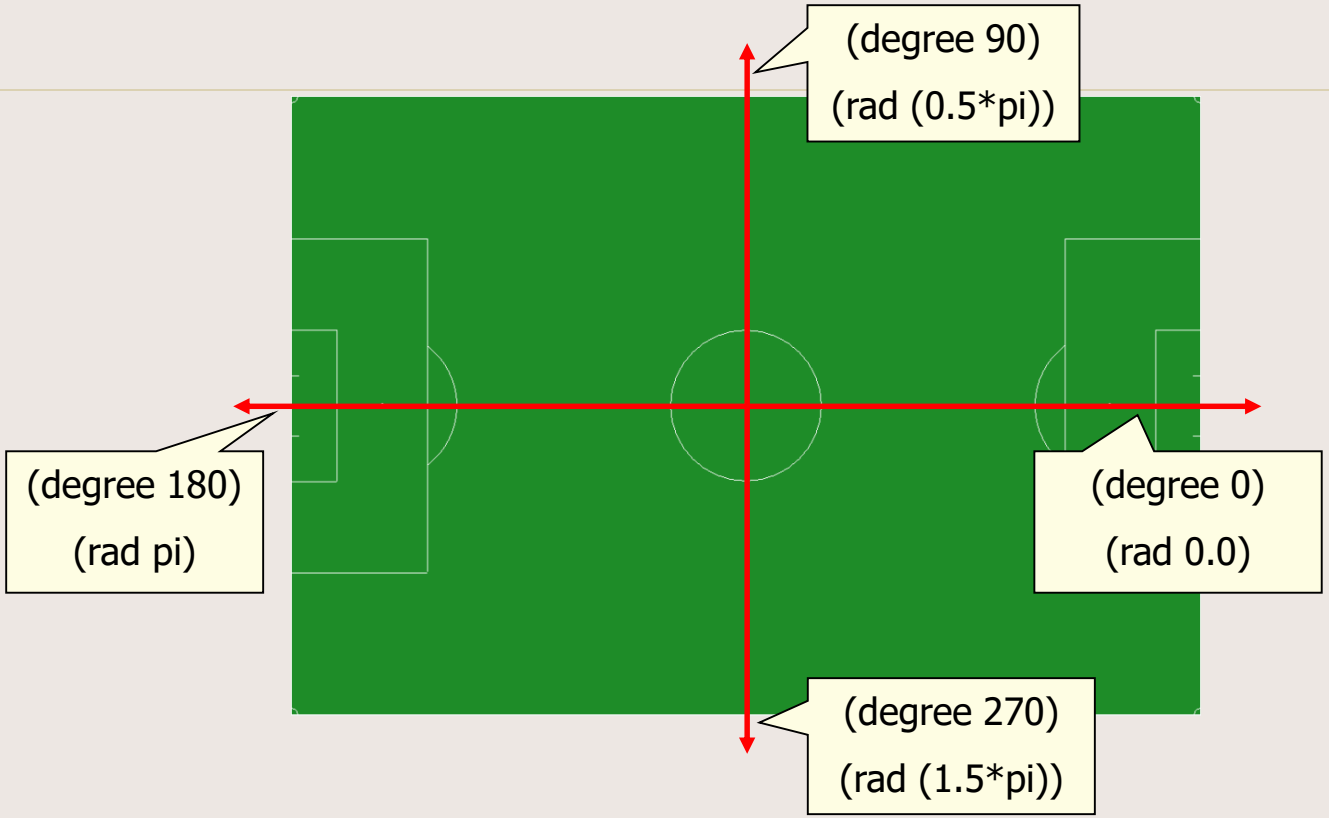
:: Position = { px :: Metre, py :: Metre }
:: Footballer = E.m: { … pos :: Position … }

Dimensions – height




```
:: Position3D    = { pxy        :: Position, pz :: Metre}
:: Football      = { ballPos    :: Position3D
                   , ballSpeed  :: Speed3D
                   }
```

Dimensions – directions

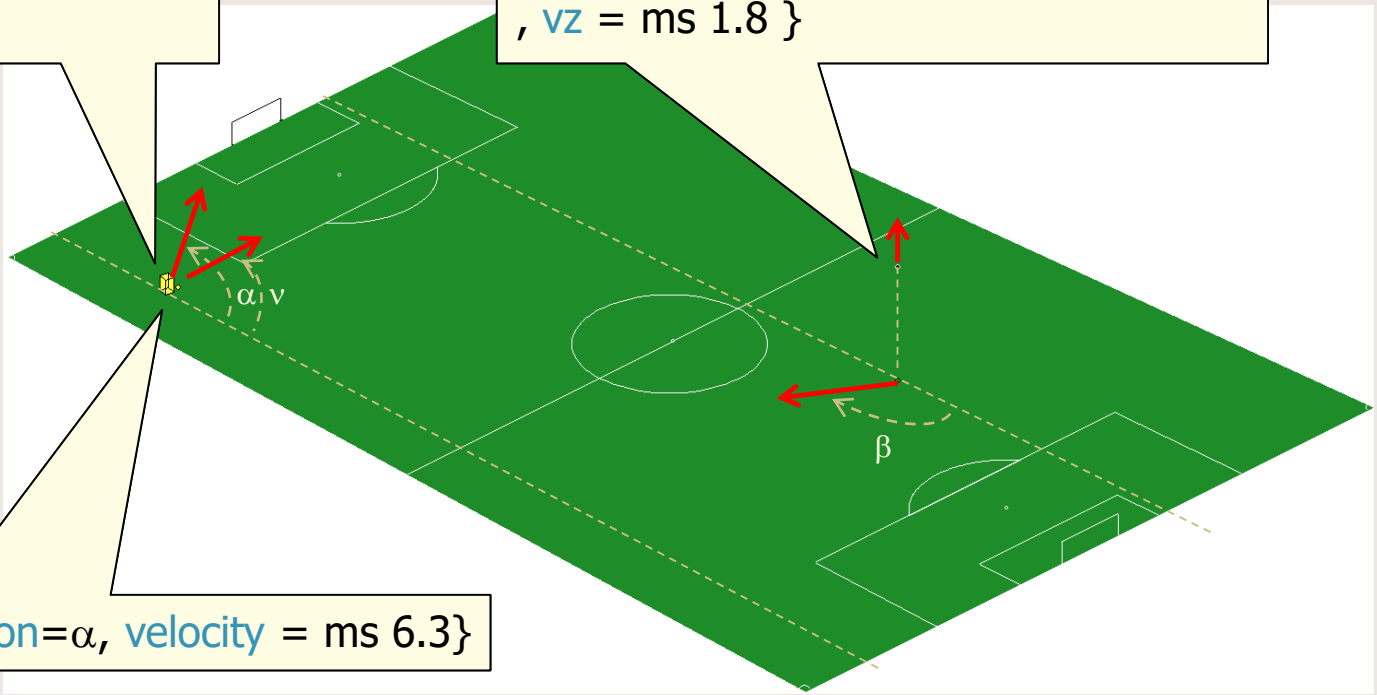



```
:: Angle
degree  :: Int   -> Angle
rad     :: Real -> Angle
pi      :== 3.14159265358979323 84
```

Dimensions – speed

… me.nose = ν …

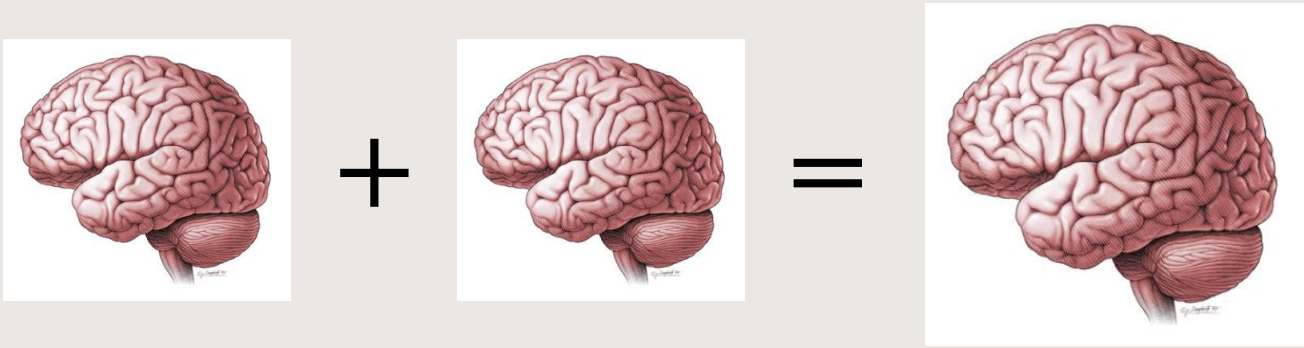
{vxy={direction=β, velocity = ms 5.5}
, vz = ms 1.8 }



{direction=α, velocity = ms 6.3}

:: Speed = {direction :: Angle, velocity :: Velocity }
:: Speed3D = {vxy :: Speed, vz :: Velocity }
:: Velocity
ms :: Real -> Velocity

Program brains



“Our” brain is different

```
:: Footballer          = E.m:
                         { …
                         , brain :: Brain (FootballerAI m) m
                         }
:: Brain ai m          = { memory :: m, ai :: ai }

:: FootballerAI m :==      (BrainInput,  m)
                        -> (BrainOutput,m)

myBrain ::  (BrainInput,   Memory)
            -> (BrainOutput, Memory)
```

What does a football player know?

```
:: BrainInput
   =   { referee      :: [RefereeAction]
       , football     :: FootballState
       , others       :: [Footballer]
       , me           :: Footballer
       }
:: FootballState
   =   Free Football | GainedBy FootballerID
```

What can a football player do?

```
:: BrainOutput :== FootballerAction
:: FootballerAction
   = Move Speed Angle | Feint FeintDirection
   | GainBall         | CatchBall
   | KickBall Speed3D | HeadBall Speed3D
   | Tackle FootballerID Velocity
```

Example 1: stand still

```
halt    :: (BrainInput,   m)
        -> (BrainOutput, m)
halt    (input, m)
        = (Move zero zero, m)
```

- module FootballerFunctions consists of a few miniscule brain functions

Example 2: run to a point

```
fix point diff (input=:{me}, m)
| d < diff        = halt (input, m)
| otherwise       = (move, m)
 where
    d             = dist me point
    a             = bearing zero me point
    r             = bearing me.nose me point
    move          = Move {direction=a,velocity=ms (toReal d)} r
```

Example 2: run to a point

```
afterfix then_do point diff (input=:{me}, m)
| d < diff        = then_do (input, m)
| otherwise       = (move, m)
 where
    d             = dist me point
    a             = bearing zero me point
    r             = bearing me.nose me point
    move          = Move {direction=a,velocity=ms (toReal d)} r
```

generalise **fix** with continuation action

Example 3: kick ball to point

```
kick point (input=:{me}, m)
| d <= maxKickReach me        = (kick, m)
| otherwise                   = halt (input, m)
where
    ball  = getBall input
    d     = dist me ball
    v     = dist me point
    a     = bearing zero me point
    kick  = KickBall {vxy = {direction=a,velocity=ms (toReal v)}
                     ,vz  = ms 1.0 }
```

Example 4: track ball

```
track_ball diff (input,m)
    = fix (getBall input).ballPos.pxy diff (input,m)
```

... what about the mini-effies?

```
miniFF diff (input,m)
    = afterfix trap_bal_in_doel
          (getBall input).ballPos.pxy diff (input,m)
```

Referee

- Referee observes game
- Referee decisions must be obeyed
  - resuming game
  - playing half
- Referee detect rule violations

Referee

```
:: BrainInput
   =   { referee :: [RefereeAction], ... }
```

- Violations:
  - hands, incorrect ball possession, dangerous play
- Results in reprimand / game resumption:
  - warning, yellow card, red card
  - red card dismisses player from game

Referee

:: BrainInput
= { referee :: [RefereeAction], ... }

- Game resumption:
  - free kick, throw in, goal kick, center kick, corner
  - acting team: Home (West / East)
  - passive team: violation if it does not let the acting team perform resumption

Referee

:: BrainInput
= { referee :: [RefereeAction], … }

- Signal scored goal:
  :: RefereeAction = … | Goal Home | …
  – followed by center kick of other team (other)

Referee

```
:: BrainInput
   =   { referee :: [RefereeAction], ... }
```

- Signal half of game:

```
:: RefereeAction = ... | EndHalf | ...
```

  – players must keep track of this

```
brain (input=:{referee},m=:{half2})
   = (..., {m & half2 = half2 || end1})
where end1 = any isEndHalf referee
```

Team building

Team building

- Team consists of 11 players:

```
:: Footballer   = {playerID     :: FootballerID, ...}
:: FootballerID = { clubName    :: ClubName
                  , playerNr    :: PlayersNumber
                  }
```

- Goal keeper has number 1
- Fielders have numbers 2..
- Players of team play for the same club

Team building

- Module Team has all (training)teams

  allAvailableTeams :: [Home FootballField -> Team]

- Team is informed of:
  - where they are at the first half (Home)
  - the size of the football field (FootballField)

  myteam :: Home FootballField -> Team

  :: Team :== [Footballer]

Brain training

Training 1: slalom

- Referee: RefereeCoach_Slalom
- Opponents: Opp_Slalom_W/E
- You: Student Slalom_E/W
- Module: Team_Student_Slalom_Assignment.icl

Use predictable mode for training

Training 1: slalom

slalom around your opponents

kick ball in goal opponent

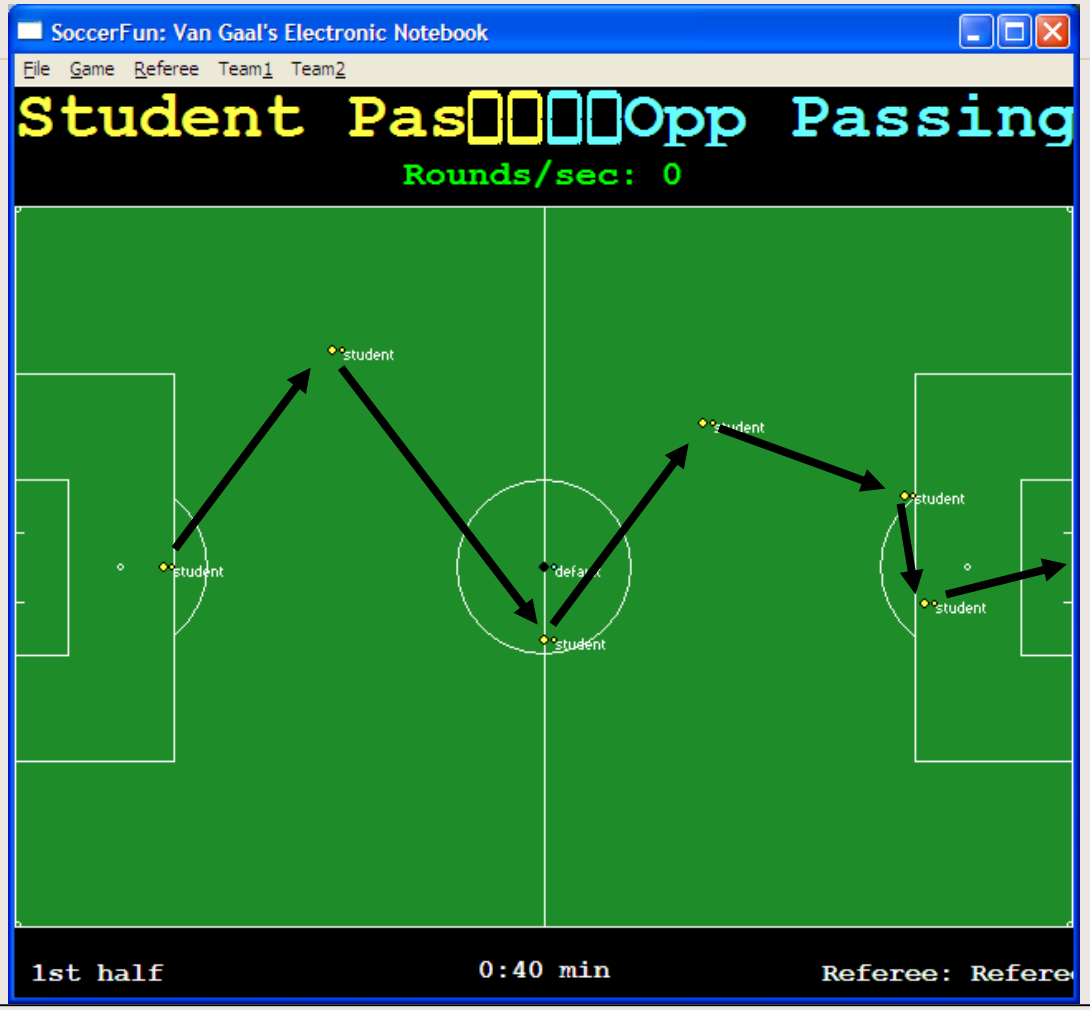
Use predictable mode for training

Training 2: play ball

- Referee: RefereeCoach Passing
- Opponents: Opp_Passing_W/E
- You: Student Passing_E/W
- Module: Team_Student_Passing_Assignment.icl

Use predictable mode for training

Training 2: play ball



Use predictable mode for training

Training 3: timing of play ball

- Referee: RefereeCoach DeepPass
- Opponents: Opp_Deep_Pass_W/E
- You: Student Deep Pass_E/W
- Module: Team_Student_DeepPass_Assignment.icl

Use predictable mode for training

Training 3: timing of play ball

Use predictable mode for training

Training 4: protect goal

- Referee: RefereeCoach Keeper
- Opponents: Opp_Keeper_W/E
- You: Student Keeper_E/W
- Module: Team_Student_Keeper_Assignment.icl

Use predictable mode for training

Training 4: protect goal

Use predictable mode for training

Read more:

- Overview Soccer-Fun: see SoccerFun\doc\quickstart_SoccerFun.pdf
- Soccer-Fun home site: http://www.cs.ru.nl/P.Achten/SoccerFun/SoccerFun.html