By Adam Duracz and Yingfu Zeng

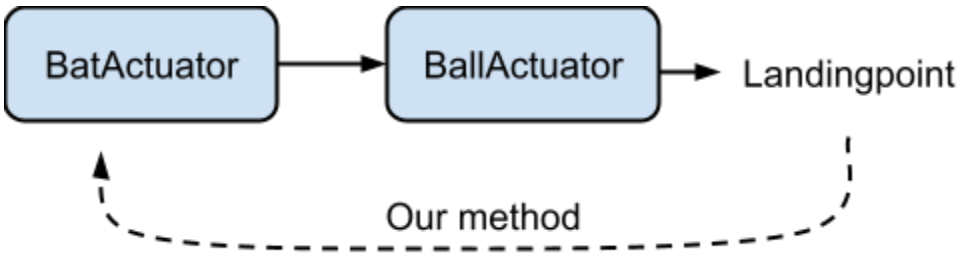
Player class

```
class Wiffwaffer(n)
 private
  mode      = "Wait";
  bounced   = false;       // Tell whether the ball bounced or not
  serve = false;           // The Game class will set serve flag to true
  hit   = false;           // when it's your turn
  count = 0;
  ballv = [0,0,0];
  ballp = [0,0,0];
  batp  = [1.6,0,0.2];
  v     = [0,0,0];         // Bat's speed
  batAngle   = [0,0,0.1];  // Normal vector of the bat's plane
  batAngle'  = [0,0,0];
  // Player(1) starts at [-1.6,0,0.2], Player(2) starts at [1.6,0,0.2]
  startPoint = [1.6*(-1)^n,0,0.2];
  v2         = [0,0,0];    // The output speed of the ball, which we desired
  v21        = [0,0,0];
  z   = 0;
  n1   = 0;
  t   = 0;
  t'  = 1;
 end
 if mode ~= "Wait" && mode ~= "Prepare" && mode ~= "Hit"
   mode = "Panic!";
 end;
 t'  [=] 1;
 switch mode
  case "Wait"               // While waiting, moving the bat to starting point
   count      = 0;
   if n == 1
     v         [=] startPoint-batp;
   else
     v         [=] startPoint-batp;
   end;
   batAngle' [=] [0,0,0]-batAngle;
   hit    = false;
   if serve == true
    mode    = "Prepare";
    bounced = false;
   else
    mode = "Wait";
   end;
  case "Prepare"             // Prepare to hit the ball
   if bounced == true        // After the ball has bounced,
                             // start moving the bat towards the ball
     v [=] (ballp-batp).*[0,20,0] + (ballp-batp).*[0,0,25] +
                (ballp+[0.12*(-1)^n,0,0]-batp).*[25,0,0];
     if norm(batp - ballp)<0.15 && abs(dot(ballp,[1,0,0])) >=
                                     0.8* abs(dot(startPoint,[1,0,0]))
      count = count+1;
      mode  = "Hit";
     end;
   end;
   // When the ball has bounced and it is at the highest position
   if count > 0 && dot(ballv,[0,0,1]) < 0.1 && bounced == true
```

```
    mode = "Hit";      // This player decide to hit.
   end;
   if dot(ballp,[0,0,1]) < 0 && bounced == false
    bounced = true;
   end;
   if(serve ~= true)
     mode = "Wait";
   end;
 case "Hit"            // Decide how you want to hit the ball,
  if n == 2
       v2  = [-(  dot(ballp,[1,0,0] ) + 0.75 + 0.5), // desired ball speed in X dim
                - dot(ballp,[0,1,0] ),               // desired ball speed in Y dim
              5 - dot(ballp,[0,0,1])];               // desired ball speed in Z dim
       v21 = (v2 - ballv) / (-2) ;
       n1   =  norm(v21);
       batAngle = v21 / n1;
       z = (n1- dot(batAngle, [1,0,0]) * 4) / dot(batAngle, [0,0,1]);
       v = ballv - [4,0,z];
  else
       v2  = [( - dot(ballp,[1,0,0] ) + 0.75 + 0.5 ), // desired ball speed in X dim
                - dot(ballp,[0,1,0] ),                // desired ball speed in Y dim
              5 - dot(ballp,[0,0,1])];                // desired ball speed in Z dim
       v21 = (v2 - ballv) / (-2) ;
       n1   =  norm(v21);
       batAngle = v21 / n1;
       z = (n1- dot(batAngle, [1,0,0])*4) / dot(batAngle, [0,0,1]);
       v = ballv - [4,0,z];
   end;
  serve  = false;
  hit    = true;
  mode   = "Wait";
 case "Panic!"
 end
end
```

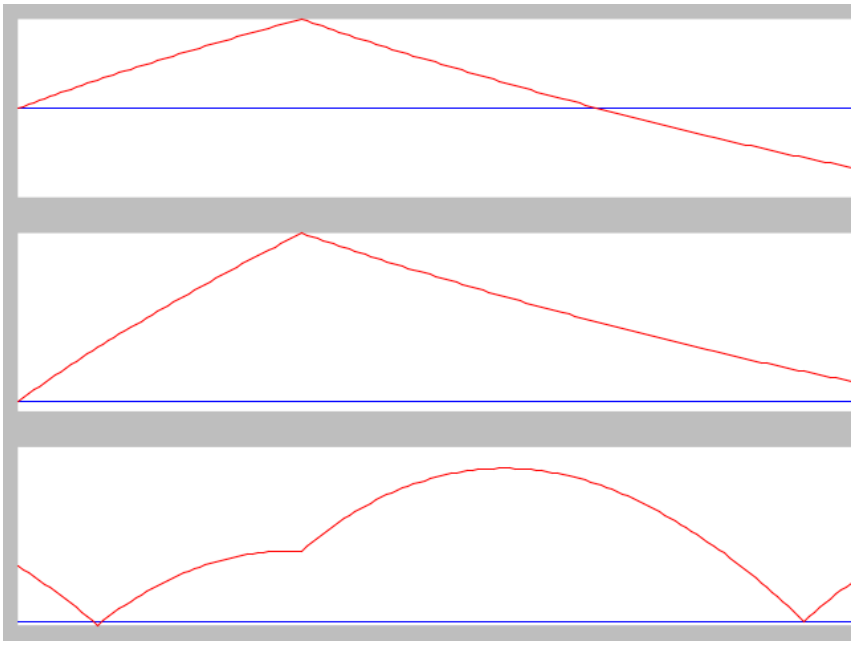
Player class design description

The design of our player is based on an approximation of the desired strategy, which is to decide on a certain desired landing point on the opponents half of the court, and then to derive the velocity and angle of the bat necessary to make the ball go there.



To simplify calculations we have assumed that the ball should spend one second in the air. Further, we approximate the force exerted on the ball by air resistance by the constant 1, which we arrived at in the following manner.
Our model assumes that air resistance is the only force acting on the ball in the X and Y directions when flying (when it is not bouncing). If we assume that the velocity of the ball in the

X direction is 3, and we want to make the ball fly for approximately 1 second every time we hit it, then the force exerted by air resistance at first is $c_{air} * v^2 = \frac{1}{6} * 3^2 = \frac{3}{2}$, which will decrease during the flight, and we approximate the average force to be 1. We use this to approximate the effect that air resistance has on the distance that the ball will travel, by integrating this constant: $\int_0^1 (\int_0^t 1 ds) dt = \frac{1}{2}$. For example, if the speed in the X direction is 3, we used Acumen to estimate how far the ball will travel with air resistance and then compared this distance to that which our approximation yields.
As can be seen in the following plot of the beginning of a game featuring our player (around the time of the first return by our player), the time elapsed between when our player hits the ball and when the ball hits the table is approximately 1 second, and the landing point is approximately $[-0.72, 0.07, 0]$ which is very close to our goal $[-0.75, 0, 0]$.



We now know the desired velocity of the ball after the impact. In order to achieve this speed we need to calculate the velocity and angle of the bat that will have this effect on the ball. The BallActuator class uses the following equation to calculate the velocity of the ball after the impact:

$$v2 = v1 - dot(2.*(v1 - v3), angle) * angle \quad (1)$$

So, we need to provide $v3$ and $angle$. We start with $angle$:

$$v2 - v1 = dot(2.*(v1 - v3), angle) * angle$$

From this equation, we can see that $dot(2 * (v1 - v3), angle)$ is a constant, so $angle = (v - v1)/norm(v2 - v1)$.

Now we need to determine $v3$, and there are many choices of this function that satisfy equation (1), we chose the vector $v1 - v3 = [4, 0, z]$ and only vary z.