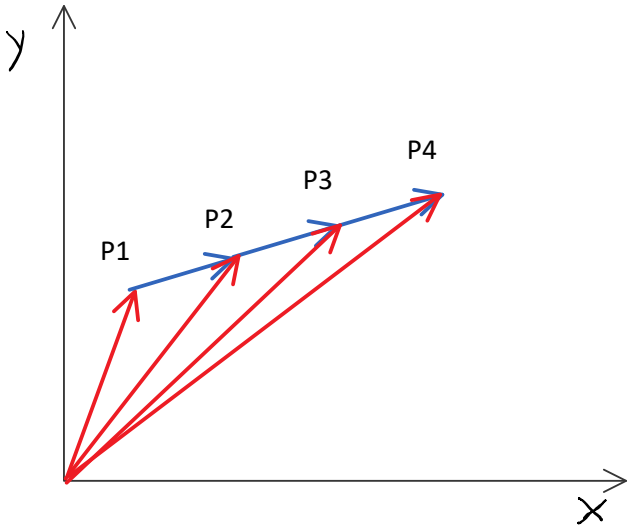
1a Ortsvektor und Geschwindigkeitsvektor fürs Geradeausfahren



Ortsvektor (wo ist das Auto)

Geschwindigkeitsvektor (wie schnell und in welche Richtung bewegt sich das Auto)

$$\vec{P}_{n+1} = \vec{P}_n + \vec{V}$$

Die Position zum Simulationszeitpunkt (n+1) ist
die Position zum Simulationszeitpunkt (n) + der Geschwindigkeitsvektor

ToTo:
Diese Skizze für den Fall, daß der Roboter eine Kurve fährt.
D.h. der Geschwindigkeitsvektor rotiert mit der Winkelgeschwindigkeit **w**

Kinematik
keine Massen

Dynamik
m, v, s, Gravitation, Reibung

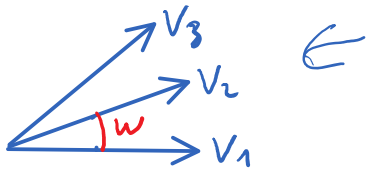
1b Ortsvektor und Geschwindigkeitsvektor Kurvenfahrt

Verglichen mit dem vorhergehenden Fall dreht sich nun der Geschwindigkeitsvektor mit der Winkelgeschwindigkeit **w**

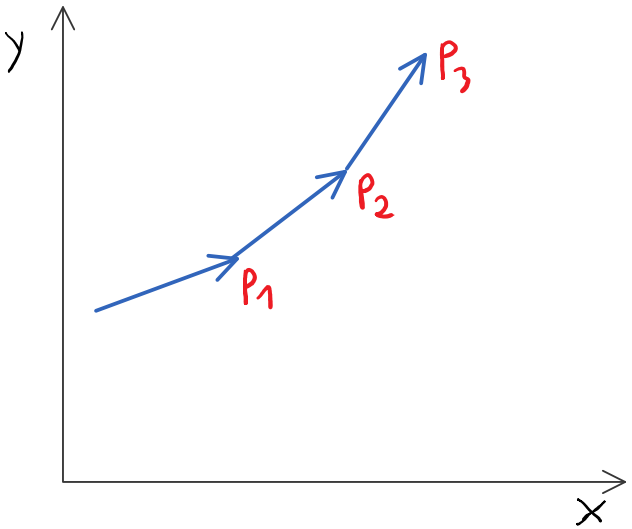
→ Ortsvektor (wo ist das Auto)

→ Geschwindigkeitsvektor (wie schnell und in welche Richtung bewegt sich das Auto)

$$\vec{V}_{n+1} = \vec{V}_n \text{ rot } w$$

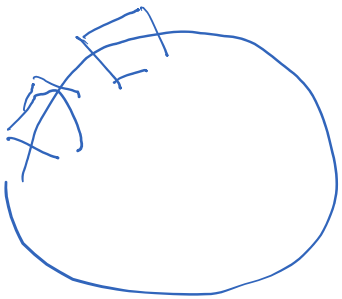


$$\vec{P}_{n+1} = \vec{P}_n + \vec{V}$$



Die Geschwindigkeit zumSimulationszeitpunkt **(n+1)** ist
Die Geschwindigkeit zum Simulationszeitpunkt **(n)** gedreht um **w**

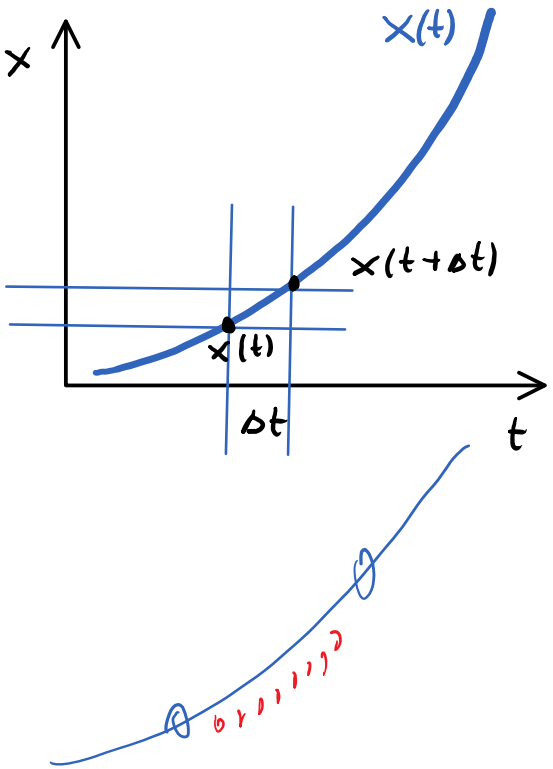
Die Position zum Simulationszeitpunkt **(n+1)** ist
die Position zum Simulationszeitpunkt **(n)** + der Geschwindigkeitsvektor



$w = dy$

2 Gleichungen für Beschleunigte (v(t)) und für konstane (v=const) Bewegung

Bewegungsgleichung für beschleunigte Bewegung
beschleunigt heißt v(t) ist nicht konstant



Bei der beschleunigten Bewegung ist x(t) eine Kurve
Bei V=const ist x(t) . . . . .

$$V(t) = \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

Momentangeschw. zum Zeitpunkt t

$$x(t+\Delta t) = x(t) + V(t) \cdot \Delta t$$

Simulationsprogramm rechnet in dt-Schritten
Übergang auf Iterationsschritte n, n+1, n+2 . . . .

$$x_{n+1} = x_n + V_{(n)} \cdot \Delta t$$

Allgemeine Gleichung für veränderliches V(t)

n	t
0	0
1	Δt
2	2·Δt
.	.
.	.
.	.

Übergang auf 2D-Vektorrechnung

$$\vec{s}_{n+1} = \vec{s}_n + \vec{V} \cdot \Delta t$$

$$x_{n+1} = x_n + V_x \cdot \Delta t$$

$$y_{n+1} = y_n + V_y \cdot \Delta t$$

2 Komponenten aufgeteilt

Spezialfall v=const. : $x_{n+1} = x_n + V \cdot \Delta t$ v ist nicht abhängig von n oder t

2a Die ITER_PER_TICK und dt Geschichte

$$x_{n+1} = x_n + v \cdot \Delta t$$

In allen unseren Bewegungssimulationen gibt es 2 wichtige Zeitgrößen

TIMER_INTERVAL .... das sind die Frames per Second

ITER_PER_TICK .... So oft werden die Bewegungsgleichungen pro neuem Frame durchgerechnet
Je größer ITER_PER_TICK gewählt wird desto genauer wird die Objektbewegung
(Physik) berechnet.

```
void OnTimer()
{
  for(i=0; i<ITER_PER_TICK; i++)
    CalcNextPositions();

}
```

Der Geschwindigkeitsmaßstab in unserer simulierten Welt ist *Pixel/FrameUpdate* und wird
so gewählt, daß sich die Objekte mit einer beobachtbaren Geschwindigkeit über den Bildschirm bewegen.

Wenn ITER_PER_TICK=1 ist spielt das **dt** für uns eigentlich keine Rolle.
Wenn wir allerdings ITER_PER_TICK verändern wollen (Rechengenaugikeit)
so müssen wir **dt = 1/ITER_PER_TICK** setzen damit die Objekte nicht unbeabsichtigt schneller werden.

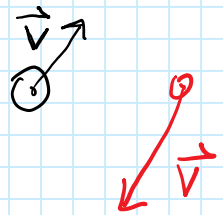
Wir werden später noch komplexere Bewegungs-Differentialgleichungen kennen lernen
(z.B. Ball mit Reibung und Gravitation)

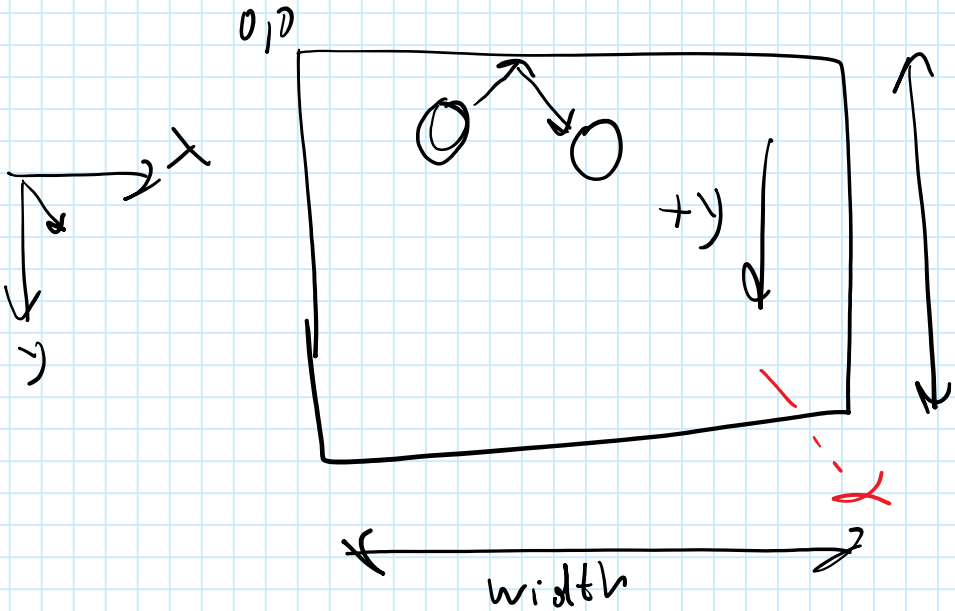
In all diesen Differentialgleichungen kommt das **dt** immer wieder vor und ist entsprechend richtig zu setzen.

Units of Measurement

Now that we are introducing mass, it's important to make a quick note about units of measurement. In the real world, things are measured in specific units. We say that two objects are 3 meters apart, the baseball is moving at a rate of 90 miles per hour, or this bowling ball has a mass of 6 kilograms. As we'll see later in this book, sometimes we will want to take real-world units into consideration. However, in this chapter, we're going to ignore them for the most part. Our units of measurement are in pixels ("These two circles are 100 pixels apart") and frames of animation ("This circle is moving at a rate of 2 pixels per frame"). In the case of mass, there isn't any unit of measurement for us to use. We're just going to make something up. In this example, we're arbitrarily picking the number 10. There is no unit of measurement, though you might enjoy inventing a unit of your own, like "1 moog" or "1 yurkle." It should also be noted that, for demonstration purposes, we'll tie mass to pixels (drawing, say, a circle with a radius of 10). This will allow us to visualize the mass of an object. In the real world, however, size does not definitely indicate mass. A small metal ball could have a much higher mass than a large balloon due to its higher density.

1 Pong 1



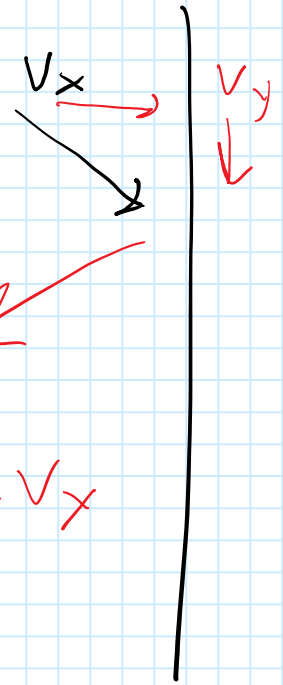


Pos. x > width false

x < 0 false

y > Heigh

y < 0



2 Pong 2

$$s(t) = v(t) \cdot \Delta t$$

$$s = v \cdot t$$

$t_0 = 0\,ms$

$t_1 = 50\,ms$ } Δt

$t_2 = 100\,ms$

Zeit kommt mit TimerEvents

$$s = v \cdot \Delta t$$ ← const

1 Dimensional ... Ball bewegt sich entlang einer Geraden

v ist in Pixel/Frame Update

$$\vec{Pos} = \begin{pmatrix} Pos.x \\ Pos.y \end{pmatrix} \qquad \vec{v} = \begin{pmatrix} v.x \\ v.y \end{pmatrix}$$

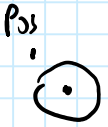
Bestimmt Betrag und Richtung der Geschwindigkeit des Balls

Bei jedem Timer-Event
wird die Ballposition neu berechnet

$$\vec{Pos}_{n+1} = \vec{Pos}_n + \vec{v} \cdot \Delta t$$

? Brauchen wir momentan nicht

$$\begin{cases} Pos.x = Pos.x + v.x \\ Pos.y = Pos.y + v.y \end{cases}$$



$$pos_{n+1} = pos_n + V_n \cdot \Delta t$$

20 Hz ≙ 50ms

→ 0.05

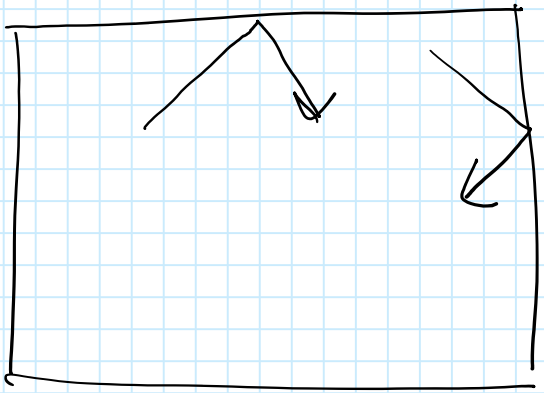
Pixel / Frame update

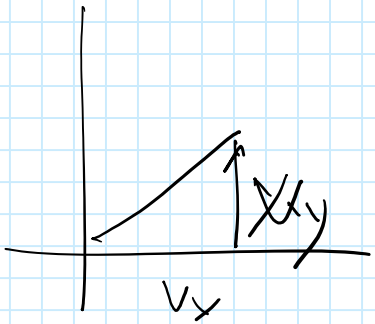
Pixel / Sec

→ Kommazahlen

$V_x = r \cdot \cos\varphi$

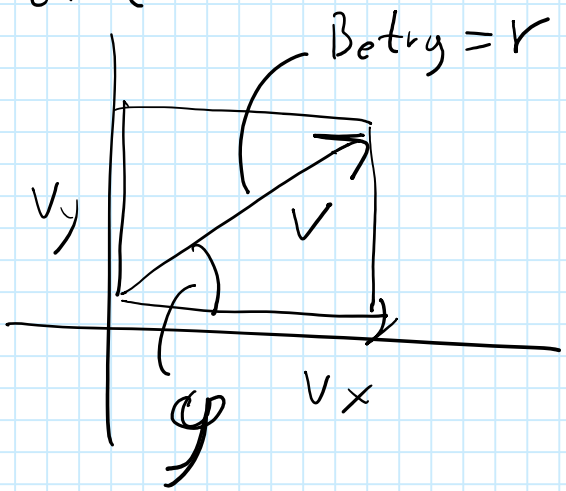
$V_y = r \cdot \sin\varphi$



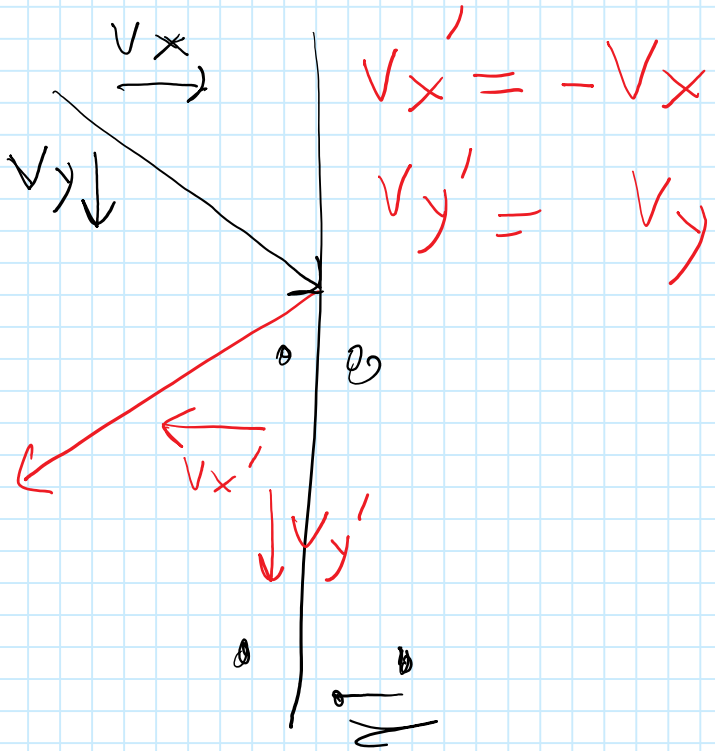


2,5

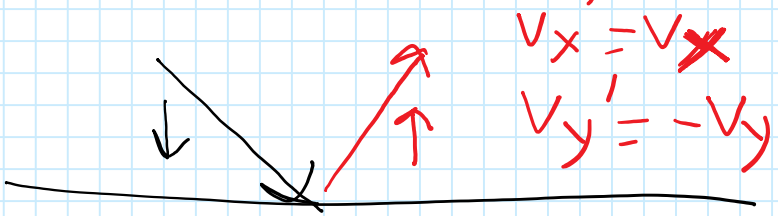
Point → PointF



```
if ( pos.x > width)
    vx = -vx;
```

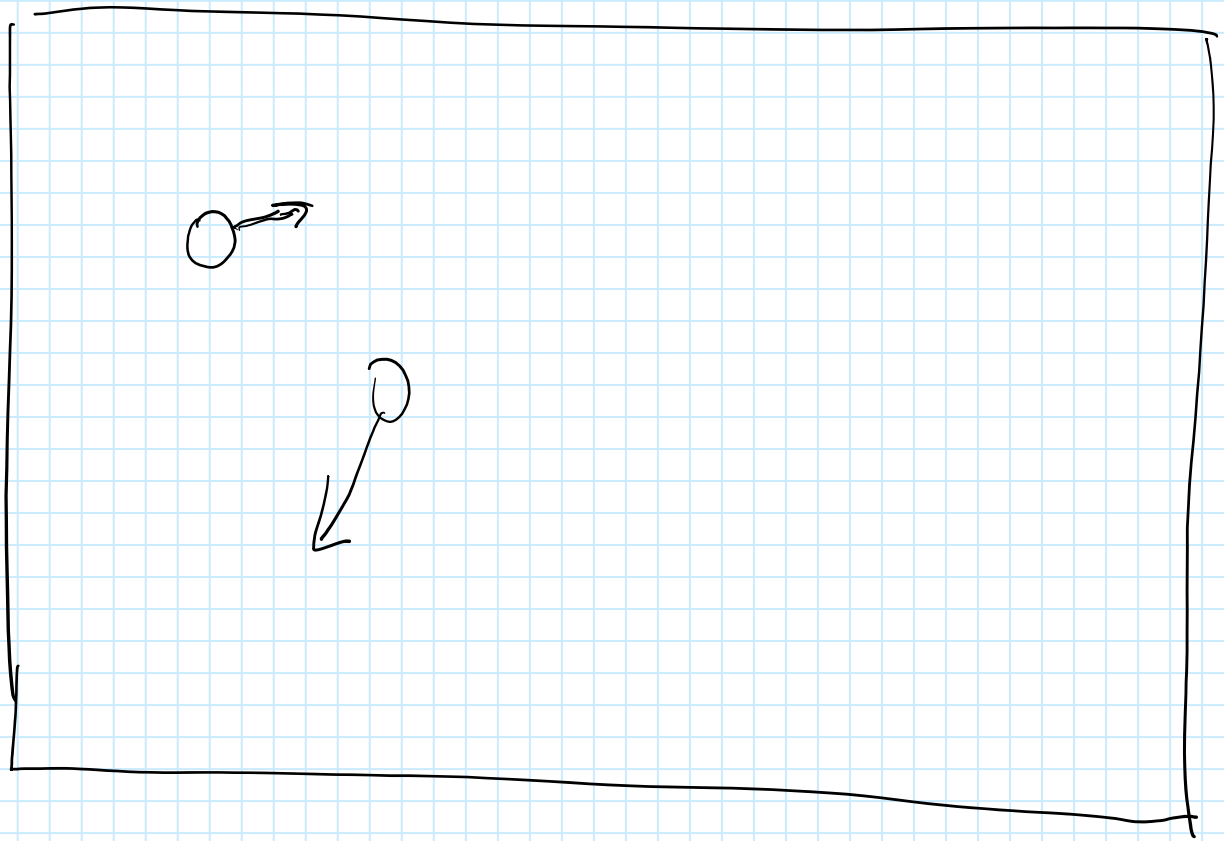



```
if ( pos.x > width && vx > 0)
    vx = -vx;
```



4 Pong 4

45Pong 5



Das Framework
gibt die FPS fix vor
20 .. 100 FPS