Lecture 3
Coordinate Systems and Trigonometry

Jan Tekülve

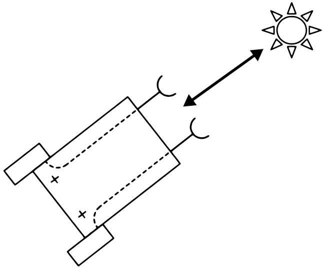
jan.tekuelve@ini.rub.de

Computer Science and Mathematics
Preparatory Course

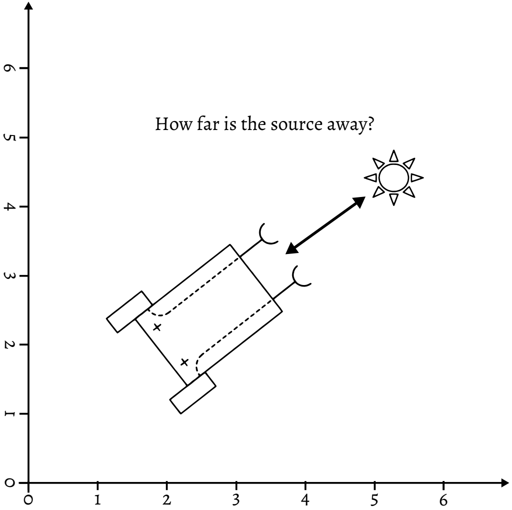
27.09.2018

Motivation - Coordinate Systems

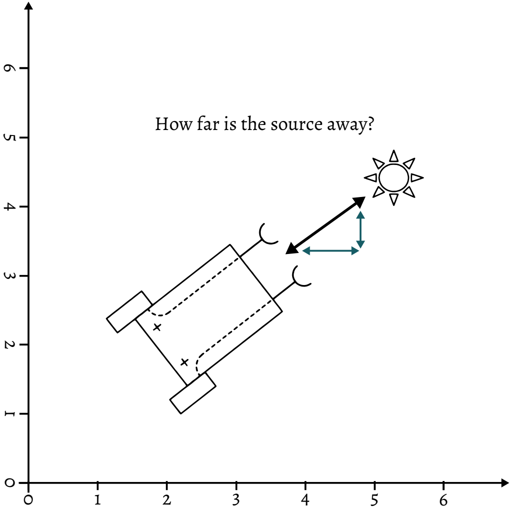
How far is the source away?



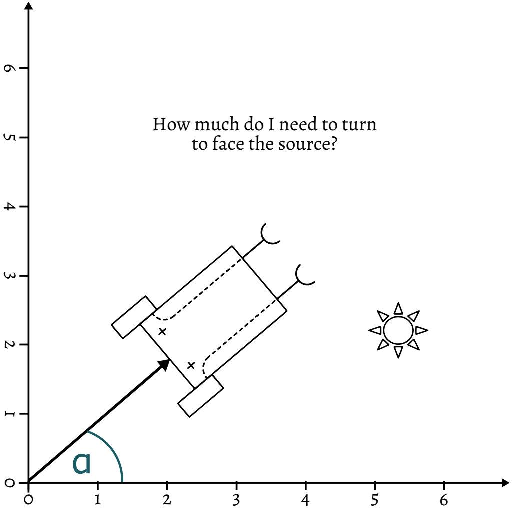
Motivation - Coordinate Systems



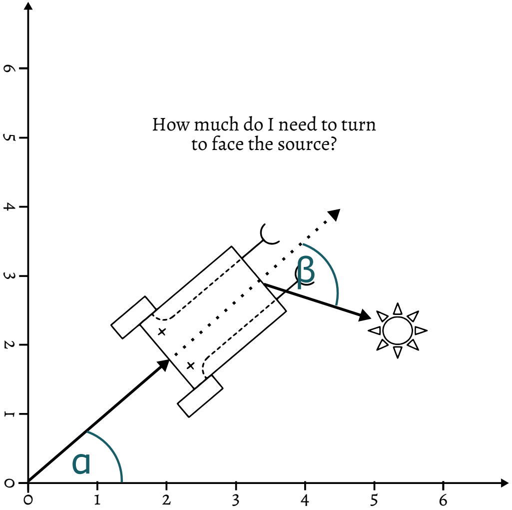
Motivation - Coordinate Systems



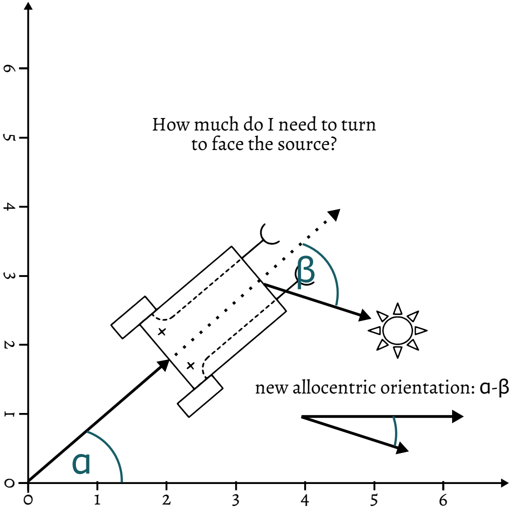
Motivation - Coordinate Systems



Motivation - Coordinate Systems

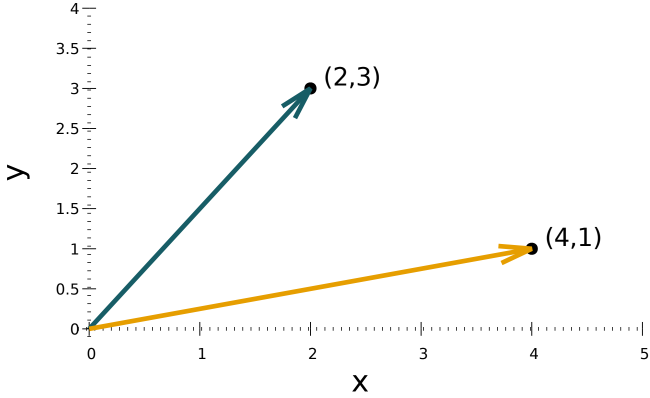


Motivation - Coordinate Systems



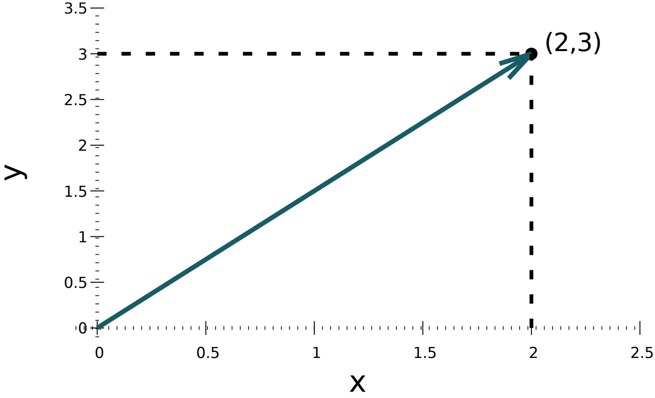
Vectors in the Cartesian Coordinate System

A vector $\boldsymbol{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ is defined as an arrow from the origin to the point (v_x, v_y)



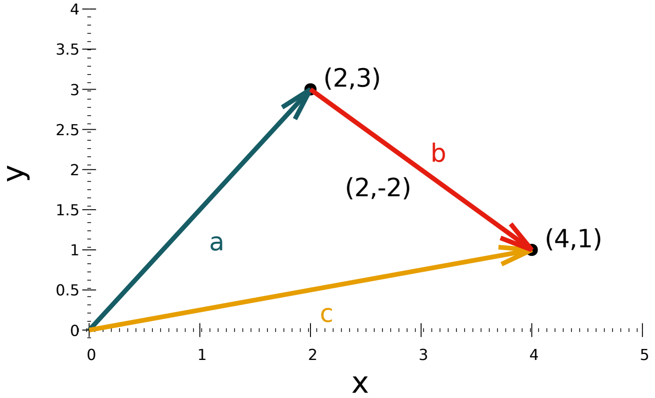
Vector Norm

The norm or length $|\boldsymbol{v}| = \sqrt{{v_x}^2 + {v_y}^2}$ of a vector $\boldsymbol{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ is calculated using the Pythagorean theorem



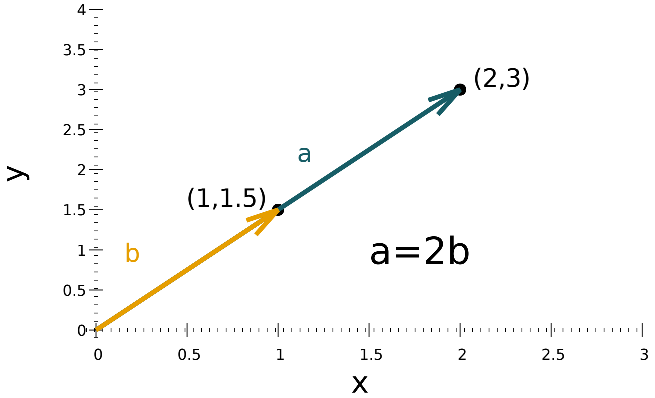
Vector Addition

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} a_x + b_x \\ a_y + b_y \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$



Scalar Multiplication

$$s\boldsymbol{a} = s\begin{pmatrix} a_x \\ a_y \end{pmatrix} = \begin{pmatrix} sa_x \\ sa_y \end{pmatrix}$$



Scalar Product

- The **scalar product** $< \boldsymbol{a}, \boldsymbol{b} >$ or $\boldsymbol{a} \cdot \boldsymbol{b}$ or $\boldsymbol{a}^T \boldsymbol{b}$ between two vectors

$$< \boldsymbol{a}, \boldsymbol{b} >=< \begin{pmatrix} a_x \\ a_y \end{pmatrix}, \begin{pmatrix} b_x \\ b_y \end{pmatrix} >= a_x b_x + a_y b_y$$

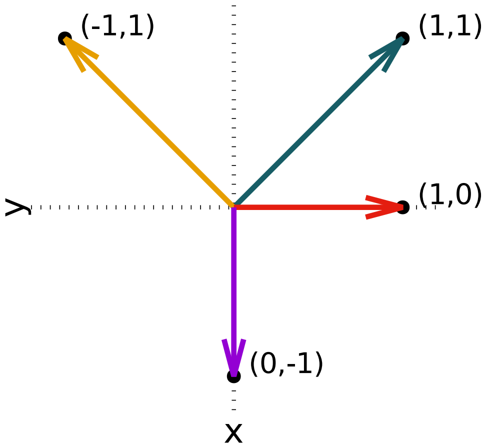
  results in a scalar value.

- It can be used to calculate an angle between two vectors:

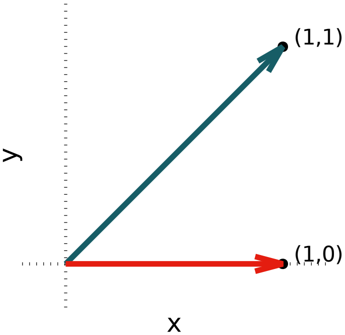
$$< \boldsymbol{a}, \boldsymbol{b} >= |\boldsymbol{a}||\boldsymbol{b}| cos(\alpha) \iff \alpha = arccos\left(\frac{< \boldsymbol{a}, \boldsymbol{b} >}{|\boldsymbol{a}||\boldsymbol{b}|}\right)$$

- If $< \boldsymbol{a}, \boldsymbol{b} >= 0$ the angle between both vectors is 90° and they are considered **orthogonal** to each other

Orthogonal Vectors



Angle between Vectors



$$\alpha = \arccos\left(\frac{< \boldsymbol{a}, \boldsymbol{b} >}{|\boldsymbol{a}||\boldsymbol{b}|}\right)$$

$$\alpha = \arccos\left(\frac{1 * 1 + 1 * 0}{\sqrt{2} * 1}\right)$$

$$\alpha = \arccos\left(\frac{1}{\sqrt{2}}\right)$$

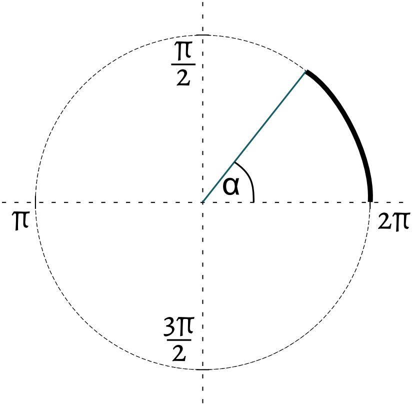
$$\alpha = \frac{\pi}{4} = 45^\circ$$

Measuring Angles

- Defining a full angle as 360° is common but actually arbitrary

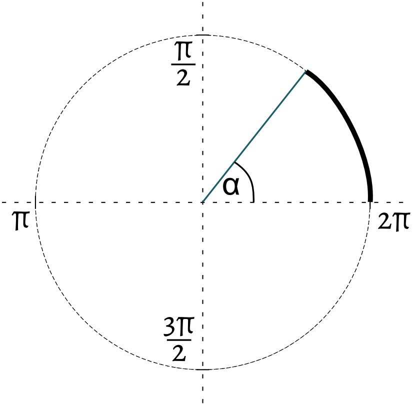
Measuring Angles

- Defining a full angle as 360° is common but actually arbitrary
- The length of the circumference of a circle is given by $2\pi r$ or 2π for the unit circle



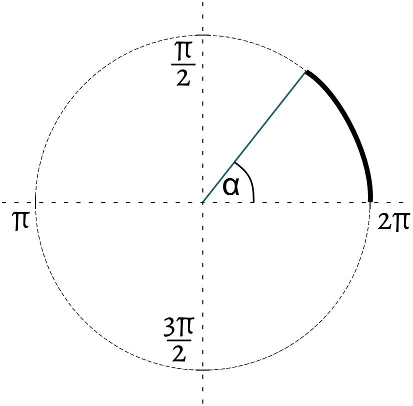
Measuring Angles

- Defining a full angle as 360° is common but actually arbitrary
- The length of the circumference of a circle is given by $2\pi r$ or 2π for the unit circle
- The length of the arc-segment enclosed by the angle is defined as **Radian**



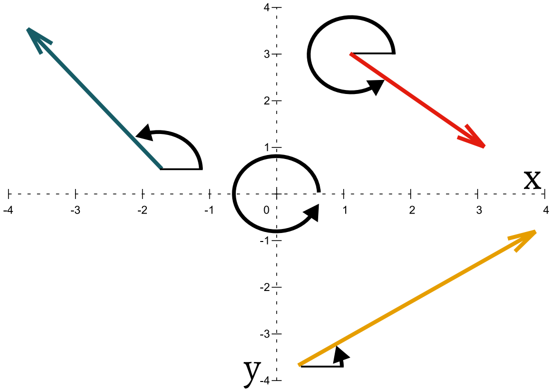
Measuring Angles

- Defining a full angle as 360° is common but actually arbitrary
- The length of the circumference of a circle is given by $2\pi r$ or 2π for the unit circle
- The length of the arc-segment enclosed by the angle is defined as **Radian**
- Calculate the Radian x from angle α in degree: $x = \frac{\alpha\pi}{180^\circ}$

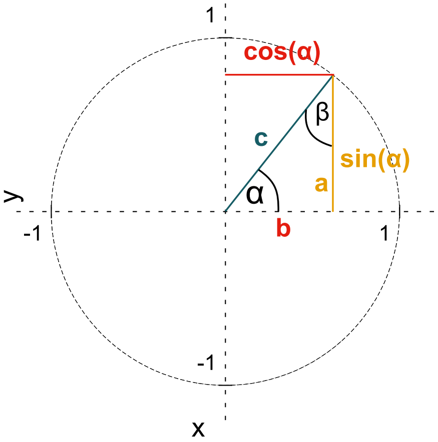


Angles in a Coordinate System

Vector orientation with respect to a coordinate system is defined by translating the origin onto the vectors tail



Calculating Angles in a Right triangle

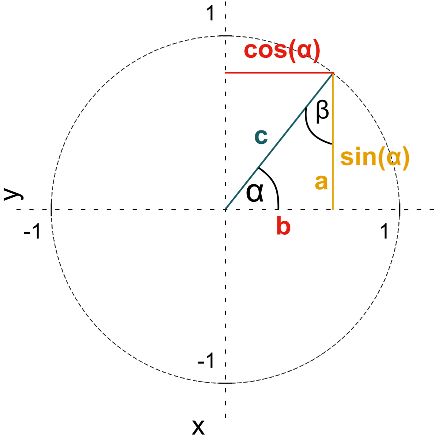


- Sine and cosine are defined in the unit circle.

$$a = cos(\alpha) \iff \alpha = cos^{-1}(a)$$
$$b = sin(\alpha) \iff \alpha = sin^{-1}(b)$$

- Click here for interactive demo.

Calculating Angles in a Right triangle

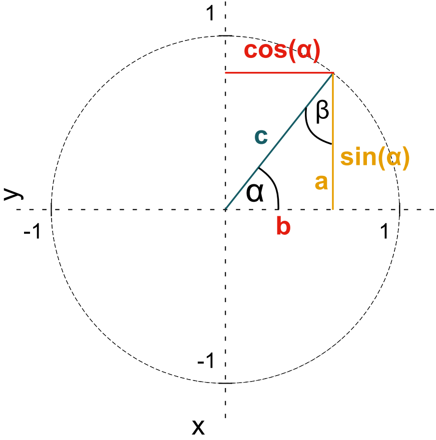


- Sine and cosine are defined in the unit circle.

$$a = cos(\alpha) \iff \alpha = cos^{-1}(a)$$
$$b = sin(\alpha) \iff \alpha = sin^{-1}(b)$$

- Click here for interactive demo.
- In the unit circle we can ignore the hypothenuse $c = 1$

Calculating Angles in a Right triangle



- Sine and cosine are defined in the unit circle.

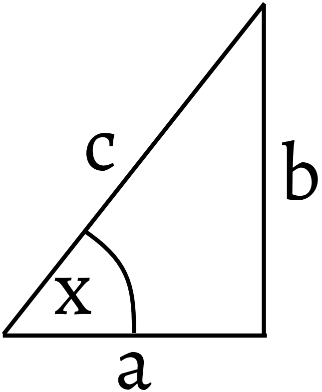
$$a = cos(\alpha) \iff \alpha = cos^{-1}(a)$$
$$b = sin(\alpha) \iff \alpha = sin^{-1}(b)$$

- Click here for interactive demo.
- In the unit circle we can ignore the hypothenuse $c = 1$
- For β the relation is reversed:

$$a = sin(\beta) \iff \beta = sin^{-1}(a)$$
$$b = cos(\beta) \iff \beta = cos^{-1}(b)$$

Rules for any Right Triangle

- $a^2 + b^2 = c^2$
- $sin(x) = \frac{b}{c} = \frac{\text{opposite}}{\text{hypothenuse}}$
- $cos(x) = \frac{a}{c} = \frac{\text{adjacent}}{\text{hypothenuse}}$
- $tan(x) = \frac{sin(x)}{cos(x)} = \frac{b}{a} = \frac{\text{opposite}}{\text{adjacent}}$



1. Motivation

2. Math
   - Vector Calculation
   - Trigonometry

3. Programming
   - Installing Python Modules
   - The Matplotlib Module
   - The Pygame Module

4. Tasks
   - Pygame Tasks
   - Pyplot Tasks

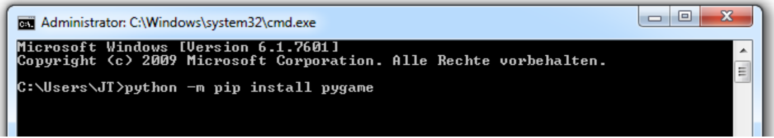
PIP installs Packages

- Pip is a helper tool that downloads and installs additional python modules. You **need an internet connection**.

- Pip can be called from the console with

```
python -m pip install <modulename>
```

- Example:



Modules for the Course

- We will need the *pygame* and *matplotlib* modules
- Make sure you have a working internet connection
- Execute the following commands one after another:

```
python -m pip install pygame
python -m pip install matplotlib
```

- A message like "Sucessfully installed ..." should be displayed after each command terminated.

The Matplotlib Module

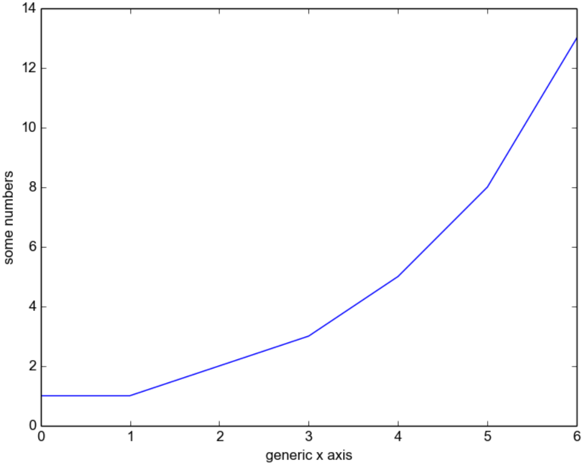
- Matplotlib is the most prominent plotting library for Python
- It was originally developed to create Matlab-like plots for free

Matplotlib.pyplot

- We will use the pyplot submodule

```
# A submodule can be imported with the . operator
import matplotlib.pyplot as plt
# The as operator allows renaming for convenience
numbers = [1,1,2,3,5,8,13]
# It is assumed that the list is a list of y-values
plt.plot(numbers)
# This generates the plot, but does not display
plt.ylabel('some numbers')
plt.xlabel('generic x axis')
plt.show()
# An alternative to showing would be to save the image
```

Result



Pyplot

- Helpful Pyplot Commands

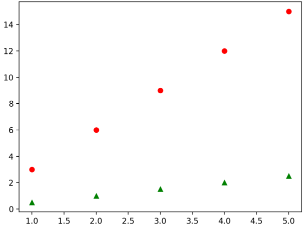
```
#Define the x and y arrays and the line appearance
#'ro' stands for red dots, 'b-' for blue lines
plt.plot([1,2,3,4], [1,4,9,16], linewidth=2.0, 'ro')
#Explicitly define the range of the axis
plt.axis([0, 6, 0, 20])
#Save the plot as an image with a desired resolution
plt.savefig('myplot.png',dpi=200)
```

- Find detailed examples here https://matplotlib.org/users/pyplot tutorial.html

Pyplot

- Multiple plots in one figure

```
x = [1,2,3,4,5]
y1 = [3,6,9,12,15]
y2 = [0.5,1,1.5,2,2.5]
plt.plot(x,y1,'ro',x,y2,'g^')
```



The Pygame module

- Pygame contains a set of modules designed for video game writing
- It is an open-source project since 2000, latest update 2017
- Its classes allow high-level game programming

Setting up an environment

- Every pygame script should contain this

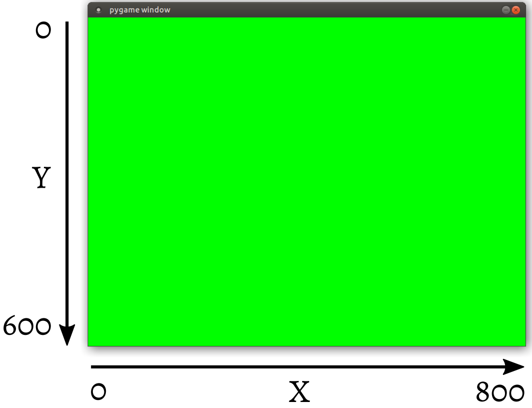
```
import pygame, sys #Import pygame and system functions
from pygame.locals import * #Import all pygame modules
pygame.init() #Initialize all modules
```

- Set up a 800x600 frame

```
#Define the frame size
frame = pygame.display.set_mode((800, 600))
#Fill the frame with a R,G,B color
green = (0,255,0)
frame.fill(green)
pygame.display.flip() #!Important! Update the display
```

Pygame Coordinate System

The Pygame coordinate System has its origin in the top left corner



The Game Loop

- A game should only end through user interaction

```
#This loop runs forever
while True:
    #pygame.event catches user interaction in a list
    for event in pygame.event.get():
            #For example a click on the close-button
            if event.type == QUIT:
                #This exits the game appropriately
                pygame.quit()
                sys.exit()
```

- For simplicity the pygame.event for-loop will be omitted in future slides

Positioning Objects

- pygame.Rect - object for storing rectangular coordinates

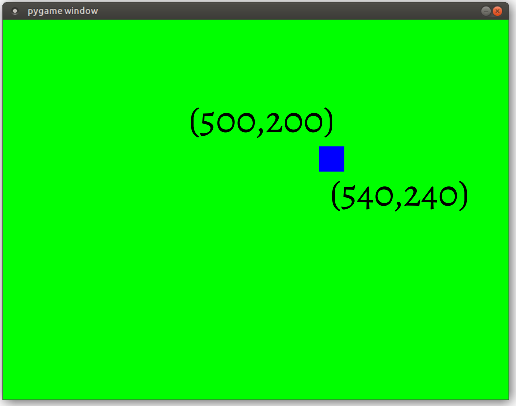
```
#pygame.Rect((left, top), (width, height))
#A square at Pos 500,200 with size 40
square = pygame.Rect((500,200),(40,40))
```

- Rects can be drawn on the screen

```
frame = pygame.display.set_mode((800, 600))
frame.fill((0,255,0))
#pygame.draw.rect(screen, color, pygame.rect)
pygame.draw.rect(frame, (0,0,255), square)
pygame.display.flip()
```

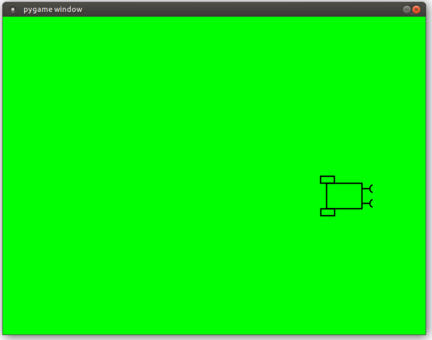
Draw Rectangle Example

```
square = pygame.Rect((500,200),(40,40))
pygame.draw.rect(frame, (0,0,255), square)
```



Loading Images

```
#Loads the Image
vehicle = pygame.image.load("braitenberg.png")
vehicle.convert() #Converts the image to game coordinates
frame.blit(vehicle,(600,300)) #Places it on the screen
```



Using the Game-Loop

- Moving the vehicle across the screen

```
# We loaded the image in vehicle
# and set up a screen in frame
xPos = 100 #Start Position
frame.blit(vehicle,(xPos,300)) #Draw the vehicle

while True:
    xPos = xPos +1 #Increase the xPos

    frame.blit(vehicle,(xPos,300)) #Draw at the new pos
    pygame.display.flip() #Show the Updates
```

Using the Game-Loop

- Moving the vehicle across the screen

```
# We loaded the image in vehicle
# and set up a screen in frame
xPos = 100 #Start Position
frame.blit(vehicle,(xPos,300)) #Draw the vehicle

while True:
    xPos = xPos +1 #Increase the xPos

    frame.blit(vehicle,(xPos,300)) #Draw at the new pos
    pygame.display.flip() #Show the Updates
```

- **This draws all vehicles on top of another!**

Using the Game-Loop

- Moving the vehicle across the screen

```
# We loaded the image in vehicle
# and set up a screen in frame
xPos = 100 #Start Position
frame.blit(vehicle,(xPos,300)) #Draw the vehicle

while True:
    xPos = xPos +1 #Increase the xPos
    frame.fill((0,255,0)) #Paint over the old canvas
    frame.blit(vehicle,(xPos,300)) #Draw at the new pos
    pygame.display.flip() #Show the Updates
```

Helpful Functions

- Pygame

```
#Introduces a pause between each game loop
pygame.time.delay(500)
#This rotates an image to angle degrees
rot_sprite = pygame.transform.rotozoom(player_image,
    ↪ angle,1)
```

- Trigonometry:

```
math.pi #The number pi
math.asin(x) # sin^-1(x)
math.acos(x) # cos^-1(x)
math.degrees(radianValue) # radian to degree
```

Pygame Task Template

Explain Task Template!

Pygame Tasks

Download the files *task_3_1_template.py* and *braitenberg.png* from the course website and put them in the same folder.

1. Familiarize yourself with the template and execute it. Vary the returned angle in *calculate_angle_to_target* and verify how the vehicle turns.

2. Fill in the missing code in the function *calculate_angle_to_target*, which calculates the angle between the player position and a given target.

    - Start with a piece of paper first. Draw a triangle between the vehicle and target position. Which angle of the triangle resembles the desired vehicle orientation and how can you calculate it?
    - Make drawings estimate formulas for each of the four different cases of target positions
    - Try your solution in the code. Run the script to verify your calculations.

Pyplot Task (optional)

Take your script from the previous lecture that stores function values in a list.

1. Extend the script by also storing the x-values in a second list. Use the x and $f(x)$ list to plot your polynomial function.

2. Generate another $f(x)$ list with the same x-values, but other coefficients a_0 to a_4. Plot both functions in the same plot.

3. Save one of your plots as a '.png' image with 300 dpi. Add labels at your own discretion.