NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

MA4829

Machine Intelligence

(AI 1.0 <><> AI 2.0 <><> AI 3.0)

XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

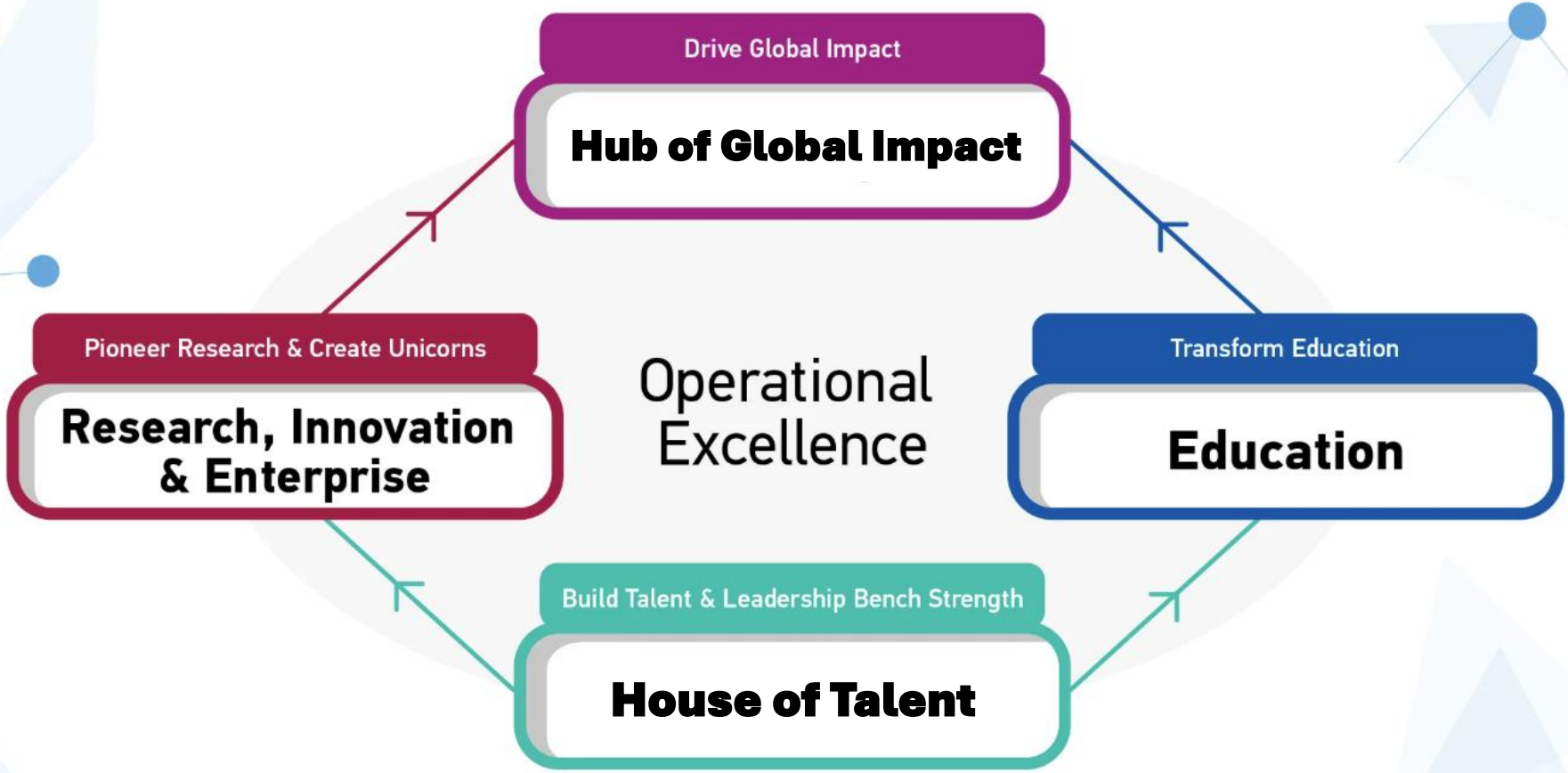
What is a system with intelligence inside?

Outline of AI 3.0

- Module 1: True Foundation of Visual Intelligence
- Module 2: Visual Knowledge Representation
- Module 3: Visual Knowledge Perception
- Module 4: Visual Knowledge Computation
- Module 5: Visual Knowledge Applications

ABOUT NTU

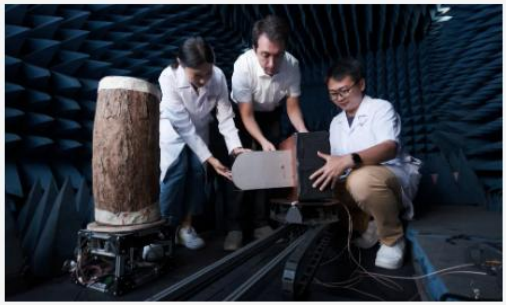
Remember NTU's Vision ...



Remember NTU's Mission ...

Education

We deliver transformative educational experiences that make our students both future- and AI-ready, so they are sought after by employers.



House of Talent

We attract, develop, and retain the very best people to drive excellence across the University.

Research, Innovation and Enterprise

We pursue breakthrough discoveries. We integrate technology and the humanities to address global challenges. We accelerate cutting edge innovation and create promising new enterprises.

Hub of Global Impact

We drive global impact in all that we do. We pursue long-lasting global partnerships with like-minded institutions across the world.

Education is to help citizens to fulfill their missions on Earth, which include: to understand the world and to improve the world …

0月18日 Friday
农历癸巳年九月十四
版／售价80￠
obao | zaobao.com
加坡报业控股

联合早报

南洋 ○ 星洲

部分时间托管残疾者
明年也可获80%津贴
3 焦点

拒载德士增加
今年210人受罚 4 新加坡

华乐团招募4000人
明年在国家体育场办音乐会
7 新加坡

李总理：排名不应是办大学终极考量

黄伟曼 报道
ngwaimun@sph.com.sg

本地大专学府的主要服务对象是新加坡学生，大学除了追求学术表现之外，也扮演重要的国家与社会角色，李显龙总理提醒本地大学不应为追求名次而忽略所肩负的重任，必须持续改善本地教育，为新加坡人提供精益求精的教育。

李总理昨晚主持新加坡国立大学大学城（University Town，简称UTown）开幕仪式时指出，本地大学的世界排名一直受到关注，但在衡量大学的表现时，不应只看名次先后，排名也不应是办大学的终极考量。

他向现场近1000名国大学生、教师与嘉宾解释说，许多在世界大专教育排行榜上名列前茅的大国顶尖大学收生广泛，例如美国哈佛大学、斯坦福大学、英国牛津大学、剑桥大学、中国北京大学和清华大学等每年录取的学生中，只有一小部分是当地学生，而本地的情况却不同，大专学府主要收本地生，必须履行更广泛的国家任务。

他说："本地大学的责任在于发展学生的品格与社会良知、鼓励他们在求学时期建立长久的友谊与默契。大学要以灌输新加坡价值观和精神特质为出发点，教导学生取之社会用之社会的精神，让他们知道自己有责任推动新加坡前进。"

"要达到这个宗旨，每一所大学都应该建立自己独特的教育模式，不要一味复制其他大学。"

占地19公顷的大学城是国大搬到肯特岗校园现址后，最大型的一次扩展计划。设于大学城的四所寄宿型学院和为研究生所设的大学城宿舍，将为4000多名学生提供住宿，将教学融入住宿环境里，让来自不同背景和科系的学生一起生活、一起学习。

在大学城学院课程中，学生需在所住的学院里修读五个单元，包括两个写作单元和三个研讨课单元。他们也有机会聆听外国学者发表专题演讲，并参加各种不同的课外活动。

李总理昨天肯定国大为提升学生学习体验所做的努力，他认为大学城为学生营造了一个有利于相互学习的空间。他指出，大学城是国大的"新聚集点"，它将国大社群缩小，进一步拉近学生与同学和教授的关系。

他也以陈爱丽丝与彼德学院（College of Alice & Peter Tan)学生之前在"整合式课程"（Capstone Programme）下为乐龄人士制定健康饮食计划为例，表示希望看到大学城继续推行类似社会计划，鼓励学生积极参与这些关怀社会的活动。

国大校长陈祝全教授致辞时指出，除了设施与空间设计吸引人之外，大学城为国大社群以及国大的运作和成长带来了深刻且积极的影响。"大学城可说已成为国大校园的核心，在一些方面补足了之前肯特岗校园所缺少的生气。"

相关新闻刊第6页

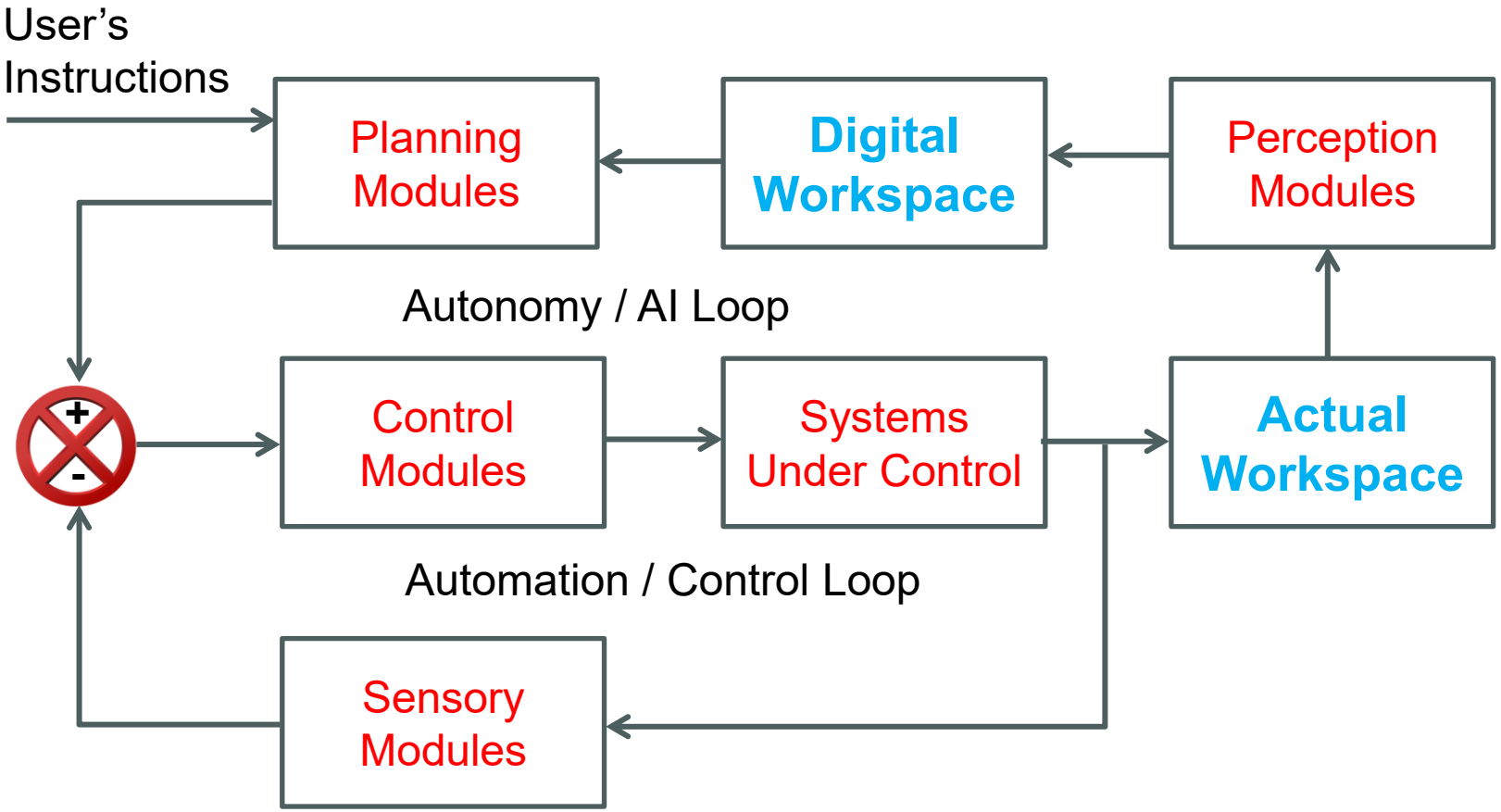
ABOUT YOU

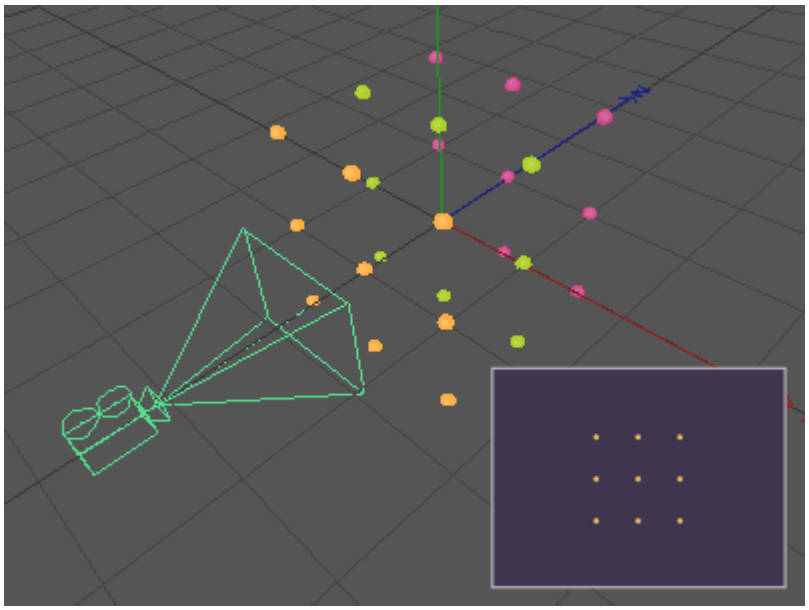
Remember your mission as MAE undergraduates …

- You are here to grow your knowledge and skills so as to be able to design machines with controllable behaviors and hopefully in some intelligent ways.

How to fulfill your mission?

- To apply learnt knowledge and skills into the implementation of the following universal blueprint underlying all the intelligent machines or systems.





ABOUT COURSE

Two Big Questions …

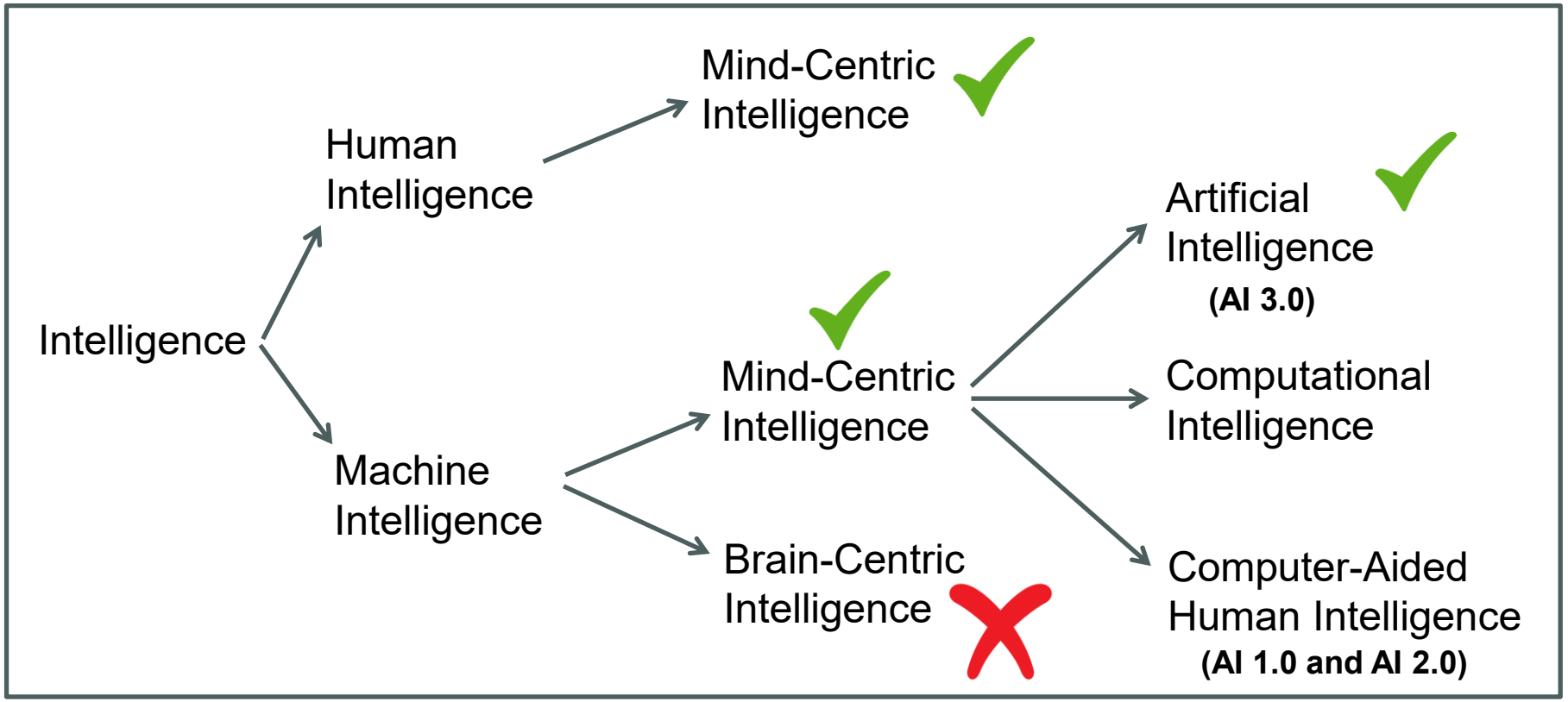
- How to achieve automation?
  - To apply the theory of control
- How to achieve autonomy?
  - To apply the theory of mind

Two Big Dilemmas …

- Top-Down Design versus Bottom-Up Optimization
  - Should the creatures in the universe be the outcomes of designs by Designers or the outcomes of natural evolution by Optimizers?

- Mind-Centric Approach versus Brain-Centric Approach
  - Does intelligence arise from Mind or from Brain?

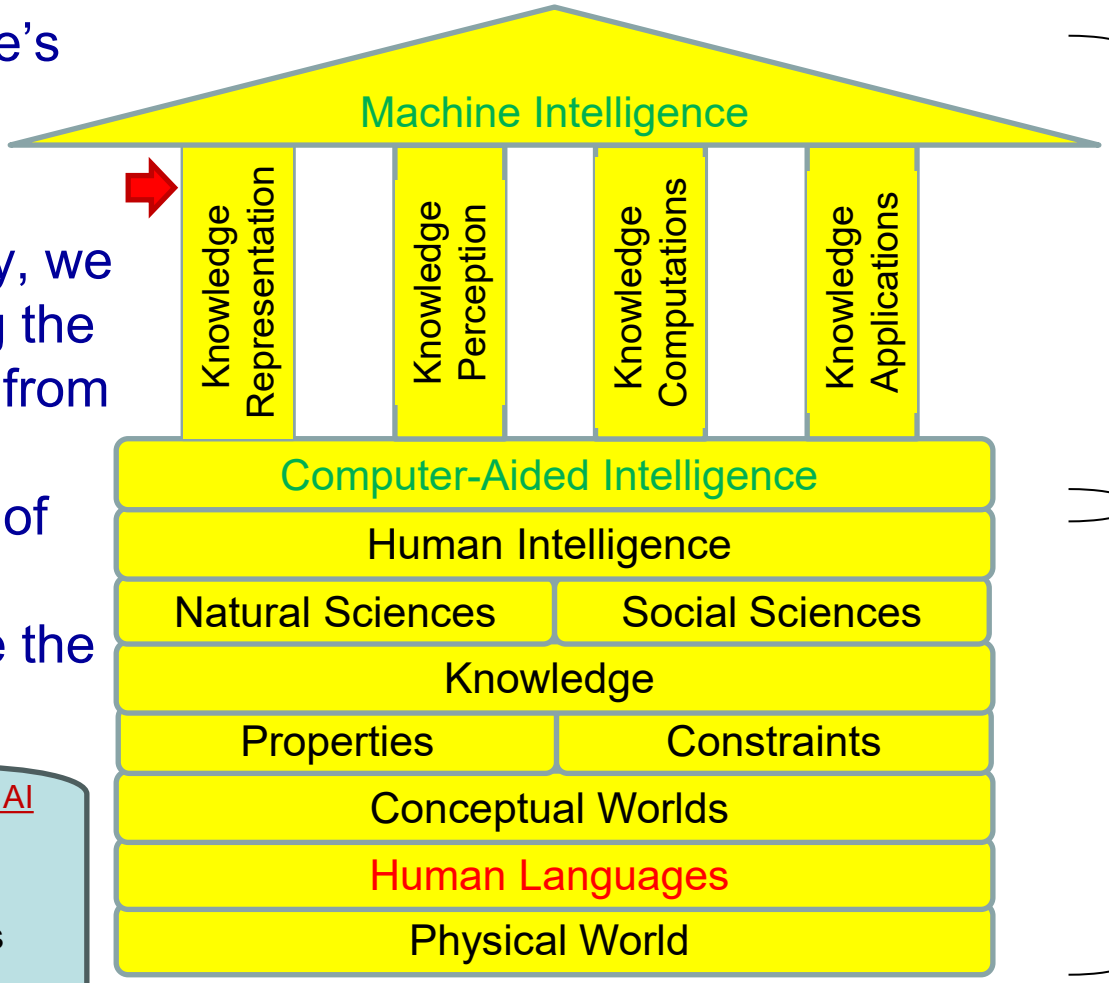
Dilemma versus Common Sense ...



Why to study visual intelligence?

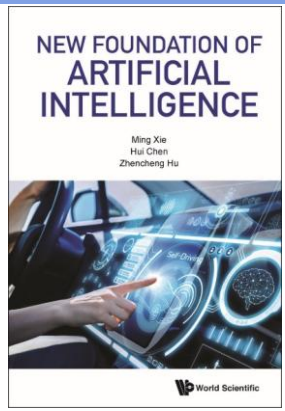
- The motivation is to achieve machine's self-intelligence.
- More specifically, we aim at achieving the transformations from visual signals to cognitive states of knowing the meanings inside the visual signals.

"We are living inside an ocean of signals"



Artificial Intelligence

Human Intelligence

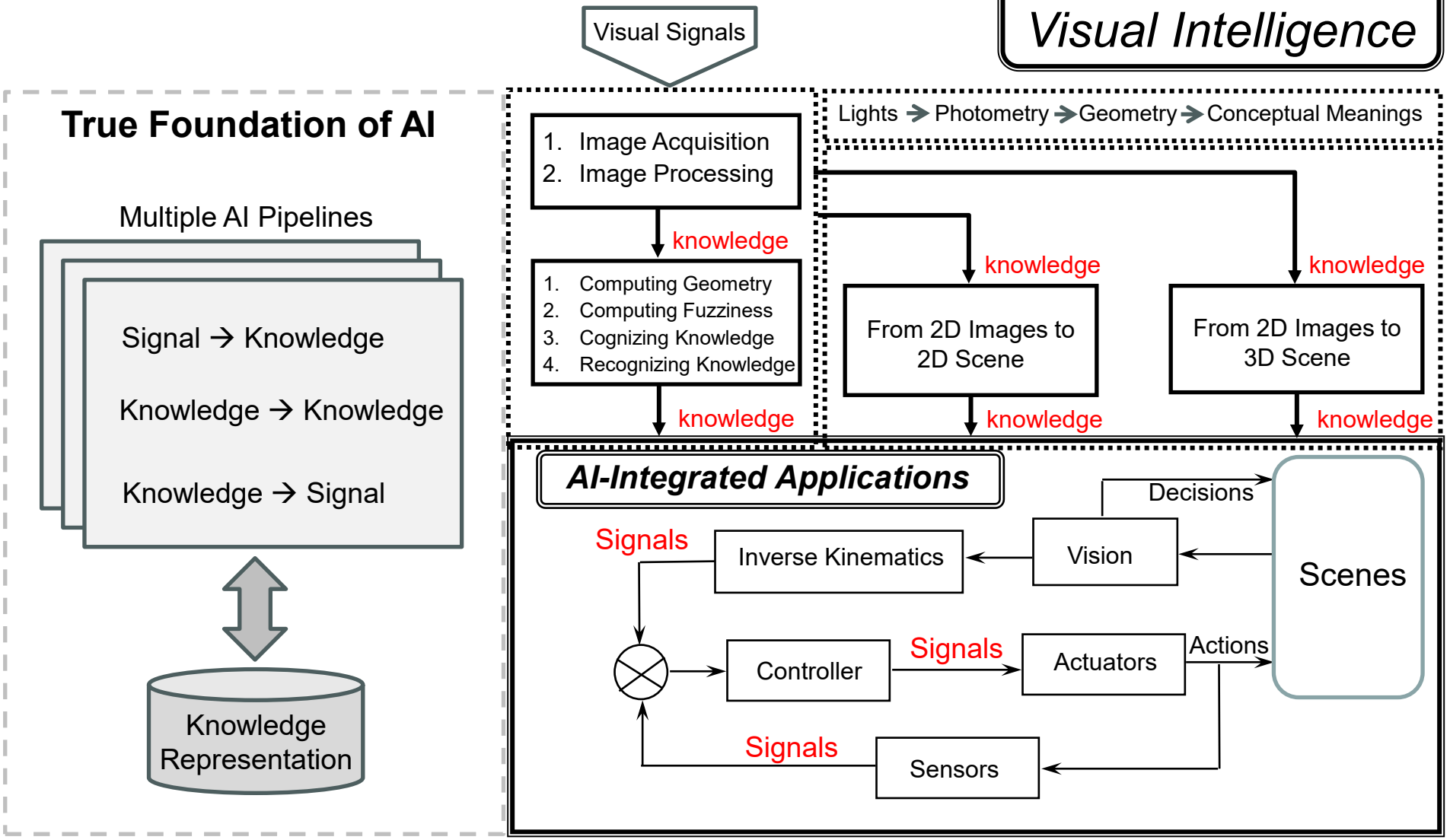


Keys of AI

- One Tool
- Two Worlds
- Three Intelligences
- Four Pillars

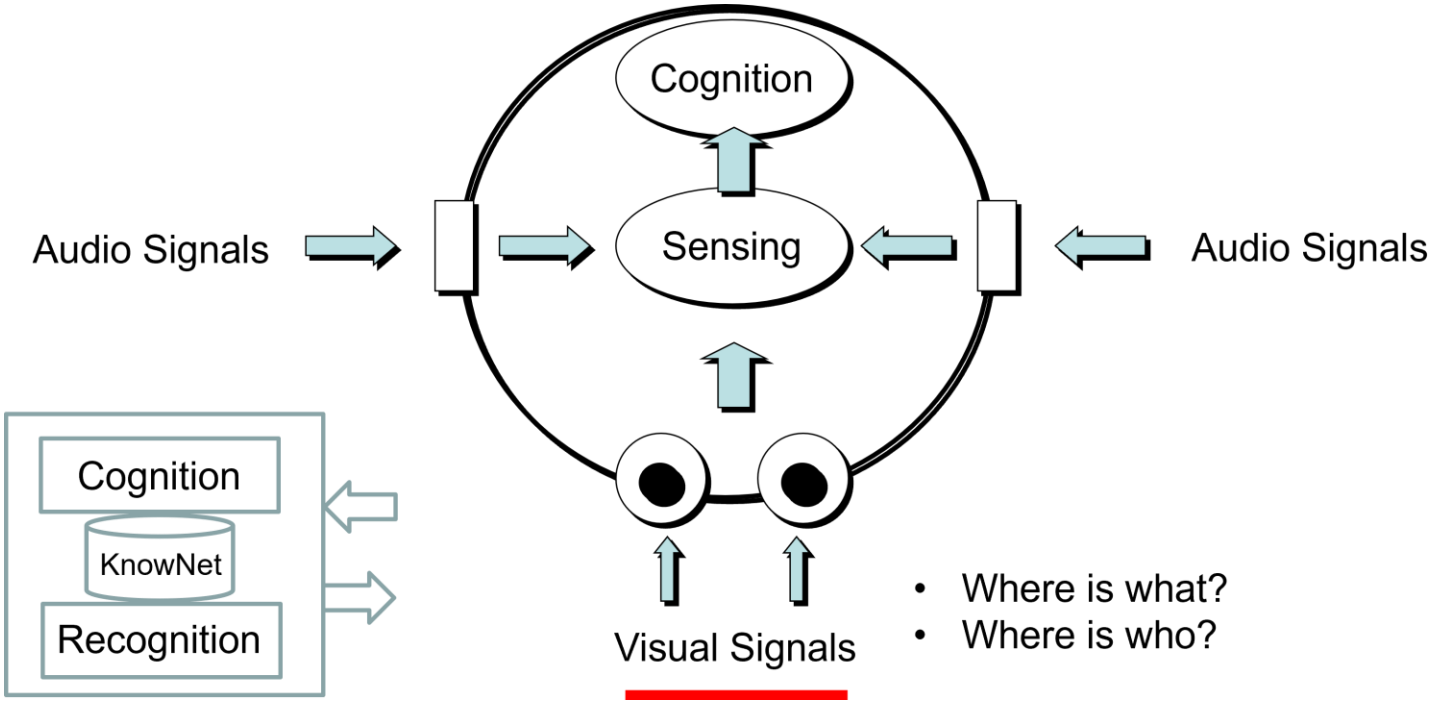
What to learn?

Roadmap of Visual Intelligence



How to study visual intelligence?

- To put yourselves into the mindset of designers of intelligent systems:
  - Who are the users? (any machine with intelligence inside)
  - What are the needs of users? (to transform visual signals into knowledge)
  - What are your intelligent systems? (to spell out the functionality)
  - What are the solutions behind the design of your intelligent systems?



How to apply?

- You could use MATLAB to quickly apply:
  - What you have read
  - What you have heard
  - What you have learnt

Practice
Practice
Practice

MATLAB = No.1 AI Software

(Rapid Prototyping Software)

Should Seeing be Believing?

We are more affected by Beliefs than by Truths.

Should Seeing Be Believing?

Terminology Alert …

1. Image properties at the lowest level include continuities and discontinuities. Hence, there are two types of pixels which are color pixels and edge pixels. These are the identities of pixels at the lowest level.
2. Therefore, "how to transform pixels into their identities?" is the so-called problem of symbol grounding in visual intelligence.
3. As a result, symbol groundings include symbol grounding of color pixels and symbol grounding of edge pixels.
4. Symbol grounding of color pixels is a process of sense-making which transforms pixels into common-sense or inner-sense of colors.
5. Symbol grounding of edge pixels is also a process of sense-making which transforms pixels into common-sense or inner sense of curves.
6. The combinations of colors and curves create meaningful patterns.

Today's Lectures …

- Module 1: True Foundation of Visual Intelligence
- Module 2: Visual Knowledge Representation
- Module 3: Visual Knowledge Perception
- Module 4: Visual Knowledge Computation
- Module 5: Visual Knowledge Applications

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Module 2

AI 3.0

MA4829 Machine Intelligence

Visual Knowledge Representation

XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

What is a system with intelligence inside?

Outline of Module 2

- Lecture 1:
  - Use of Natural Language
- Lecture 2:
  - Use of Technical Language
- Lecture 3:
  - Use of Programming Language

Outline of Module 2

- Lecture 1:
  - Use of Natural Language
- Lecture 2:
  - Use of Technical Language
- Lecture 3:
  - Use of Programming Language

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 1 of Module 2

AI 3.0

MA4829 Machine Intelligence

Use of Natural Language to Represent Knowledge

XIE Ming, PhD (France)

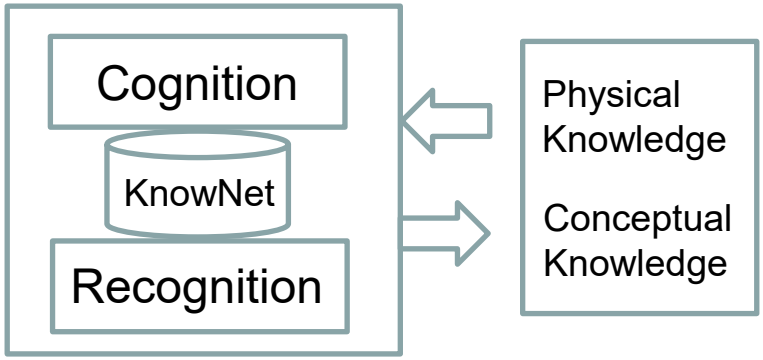
http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

What is a system with intelligence inside?

Outline of Lecture 1

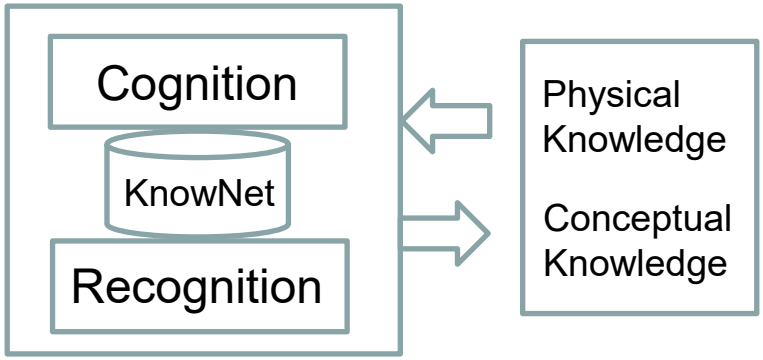
- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



- Were is what?
- Where is who?

Outline of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



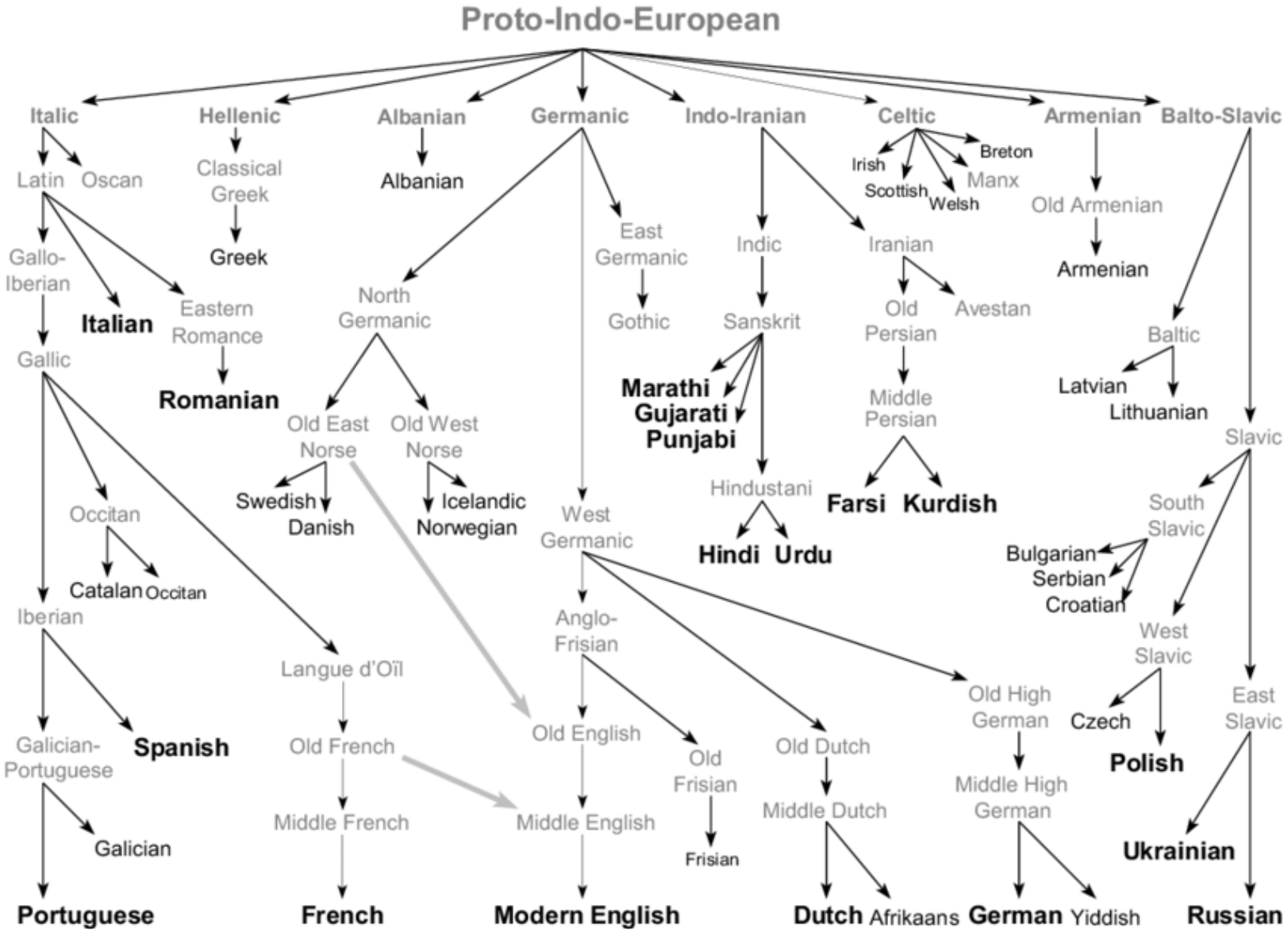
- Were is what?
- Where is who?

What is a Natural Language?

This is part of language models

- It is a set of words and rules for the purpose of projecting knowledge of the physical world into the texts which form conceptual worlds.

Evolution of Natural Languages …


Proto-Indo-European
Italic
Hellenic
Albanian
Germanic
Indo-Iranian
Celtic
Armenian
Balto-Slavic
Latin
Oscan
Classical Greek
Albanian
Irish
Breton
Scottish
Manx
Welsh
Old Armenian
Armenian
Greek
Gallo-Iberian
Italian
Eastern Romance
Romanian
North Germanic
East Germanic
Gothic
Indic
Sanskrit
Iranian
Old Persian
Avestan
Middle Persian
Marathi
Gujarati
Punjabi
Baltic
Latvian
Lithuanian
Slavic
Gallic
Old East Norse
Old West Norse
Swedish
Danish
Icelandic
Norwegian
Hindustani
Hindi
Urdu
Farsi
Kurdish
Occitan
West Germanic
South Slavic
Bulgarian
Serbian
Croatian
Catalan
Occitan
Iberian
Anglo-Frisian
West Slavic
Langue d'Oïl
Old High German
Czech
Polish
East Slavic
Galician-Portuguese
Spanish
Old French
Old English
Old Frisian
Old Dutch
Middle High German
Middle French
Middle English
Frisian
Middle Dutch
Ukrainian
Galician
Portuguese
French
Modern English
Dutch
Afrikaans
German
Yiddish
Russian

History of English Language ...

Definition of Word in English

This is part of language models

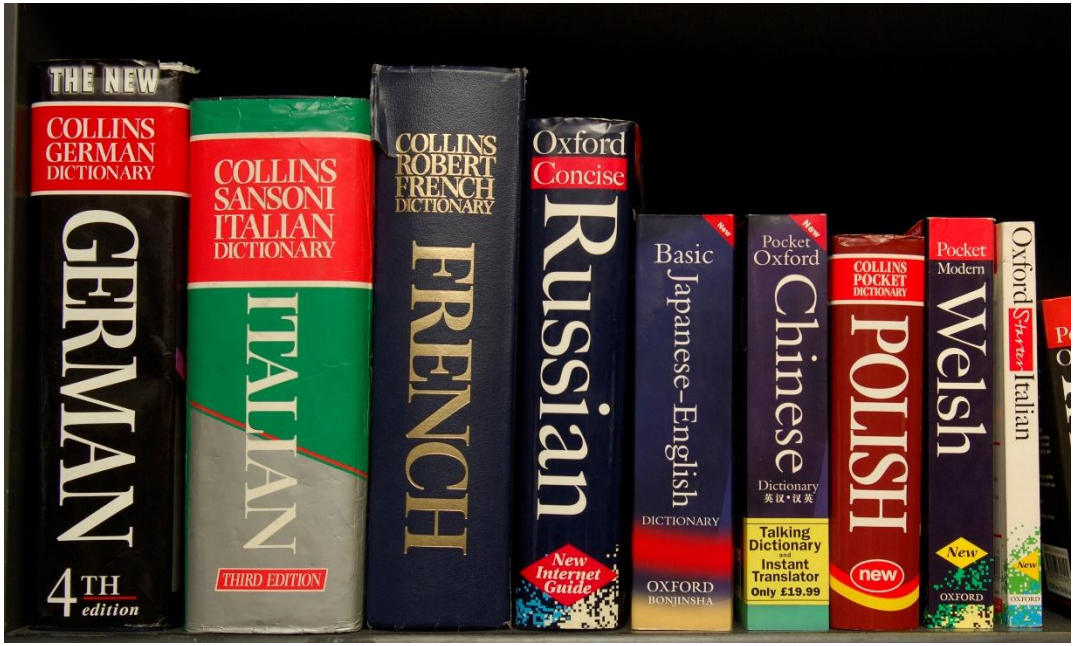
- A word is a string of characters (or strokes or others), which refers to a specific meaning in the physical world.

- Each word has:
  - A visible string of characters
  - Invisible types or constraints

Definition of Vocabulary in English

This is part of language models

- A set of words in a human language is called the vocabulary.



Definition of Grammar in English

This is part of language models

- The set of rules, which govern the properties and constraints of words in a human language, is called grammar.

- Grammar imposes:
  - Rules which constrain words
  - Rules which combine words

Types of Sentence in English

This is part of language models

- Simple sentences
  - The robot starts working at 8h30 daily.
- Compound sentences
  - Robot A is doing assembly and robot B is doing painting.
- Complex sentences
  - I look at the robot which is doing material handling.
- Requests
  - Enjoy your study! Have a nice day! Hand in the reports by 2 pm.
- Commands
  - Finish the works by 5 pm! Enter the building! Switch off the light!
- Exclamations
  - What a wonderful robot! Isn't the robot crazy!

Models of Sentence in English

This is part of language models

- Noun phrase + Verb Phrase + Noun Phrase
  - He asks Teacher for helps.

- Subject + Verb + Object 1 / Object 2 …
- Subject + Verb + Indirect Object + Direct Object
  - The teacher gives us three assignments.
  - The robot brings me a drink.

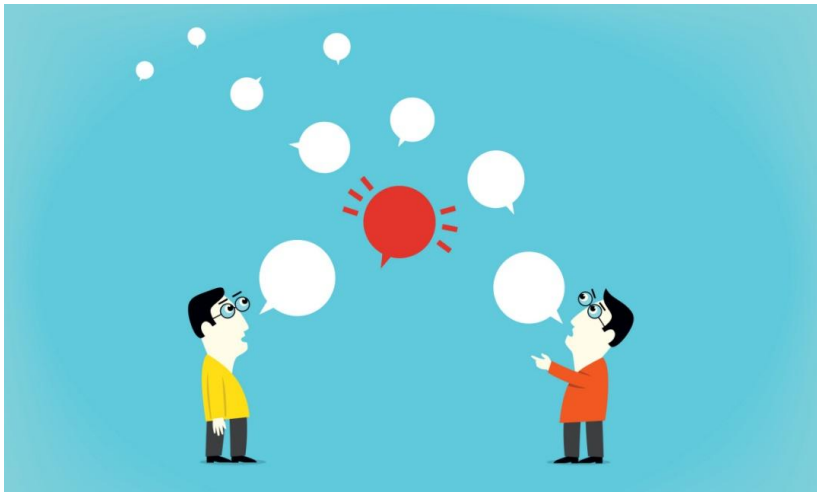
- Subject + Verb + Object + Object Complement
  - We complete the assignments timely.

Subject + Verb + Object + Adverbial (Adverb/Preposition)

He finished all the lectures happily yesterday
The robot picks up a key from a table.

Models of Questions in English

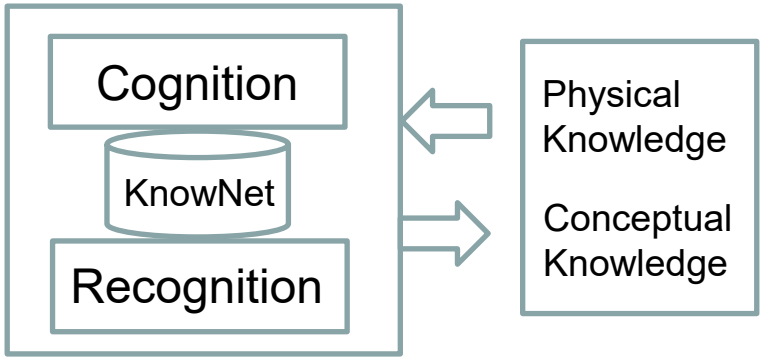
This is part of language models



- Yes/No Questions
  - Verb + Subject
    - Is this machine a robot?
    - Can a robot kill us?
- WH- Questions
  - Verb + Subject + Verb
    - Who give you that robot?
    - How many hours can the robot work?
- Tag Questions
  - Declarative sentence + Question Tag
    - A robot can do dirty jobs, isn't it?
    - He has not yet bought any book, has he?

Outline of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



- Were is what?
- Where is who?

How to represent entities?

- Use of Nouns
- Use of Noun-Phrases

Example 1

- Sky
- Red Bridge
- River
- Dark Cloud
- Mountain
- etc.

Example 2

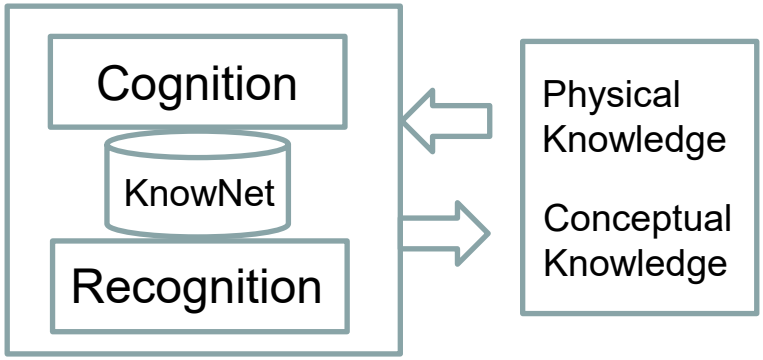
- Narrow Street
- Small Car
- Building
- Big House
- etc.

Example 3

- Night City
- Highway
- Light
- Complex Network
- etc.

Outline of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



- Were is what?
- Where is who?

How to represent systems?

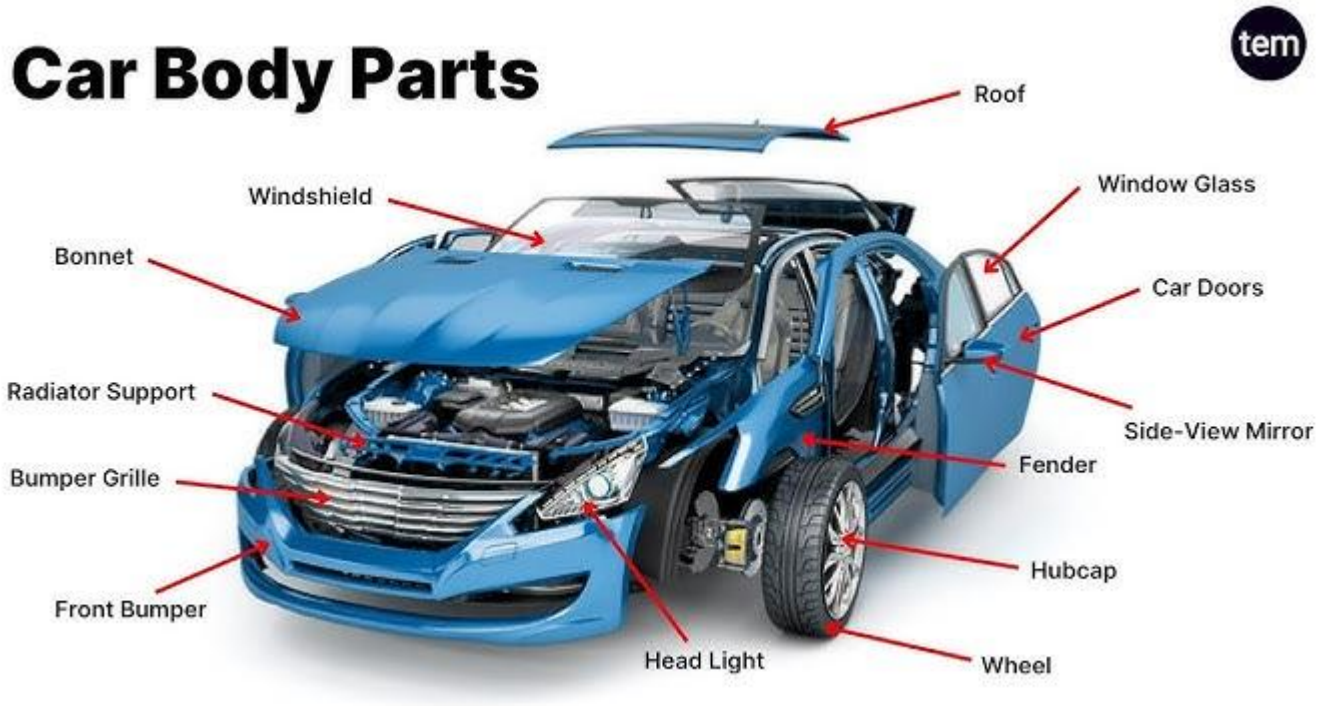
- Use of Phrases
- Use of Sentences

Example 1

- A robot manipulator is a motion system which consists of joints, links, motors, sensors and controllers, etc.

Example 2

- A vehicle is a mobile system which consists of engine, transmission, wheels, chassis, electronic control unit, sensors, lighting devices, communication devices, seats, doors, etc.

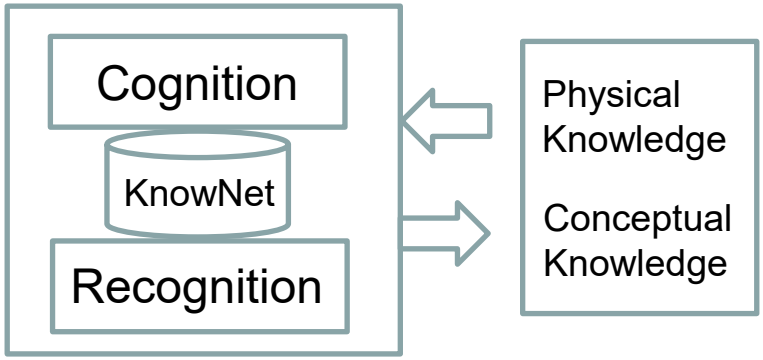


Example 3

- A production line is a material flow system which consists of conveyors, workstations, equipment, controllers, sensors, displays, communication devices, etc.

Outline of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



- Were is what?
- Where is who?

How to represent interactions?

- Use of Verbs
- Use of Verb-Phrases

Example 1

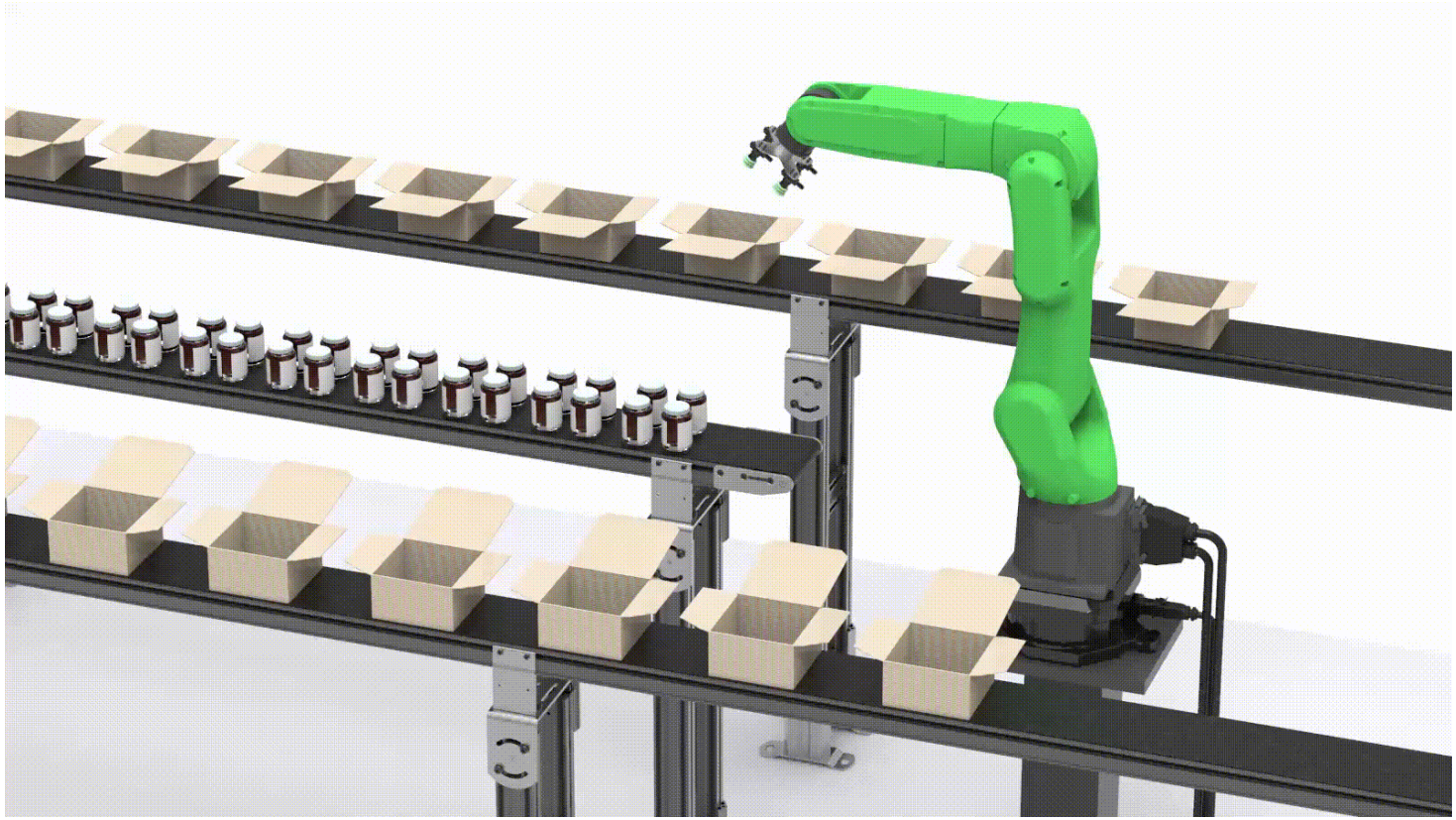
- Run
- Run fast
- Swing by Leg
- Push by Leg
- Land by Foot
- etc.

Example 2

- Squeeze
- Compress
- Flatten
- Hold
- Grasp
- etc.

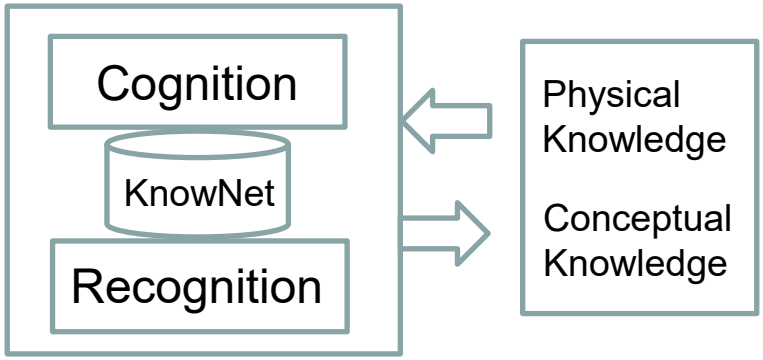
Example 3

- Pick
- Place
- Move
- Follow
- Open
- Close
- etc.



Outline of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



- Were is what?
- Where is who?

How to represent events?

- Use of Sentences
- Use of Paragraphs

Example 1

- A lady sings emotionally with a loud voices.
- A boy sings happily with eyes closed.

Example 2

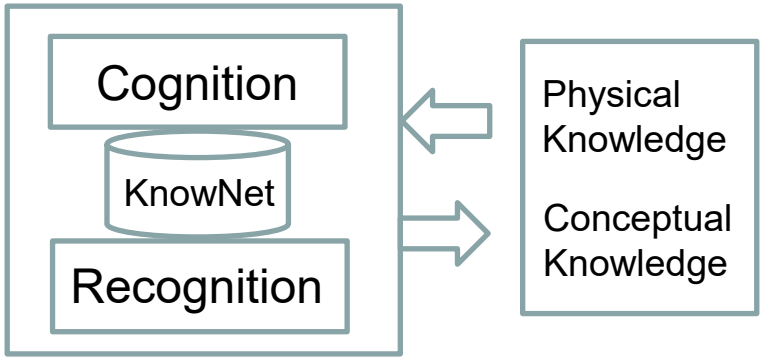
- A dancer is performing on a stage. She waves her arms and swings her legs alternatively while other dancers are standing still behind the scene.

Example 3

- A car is following a moving target. When the moving target accelerates, it also accelerates. When the moving target stops, it stops as well. When the moving target changes direction of movement, it also changes its direction of motion.

Outline of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



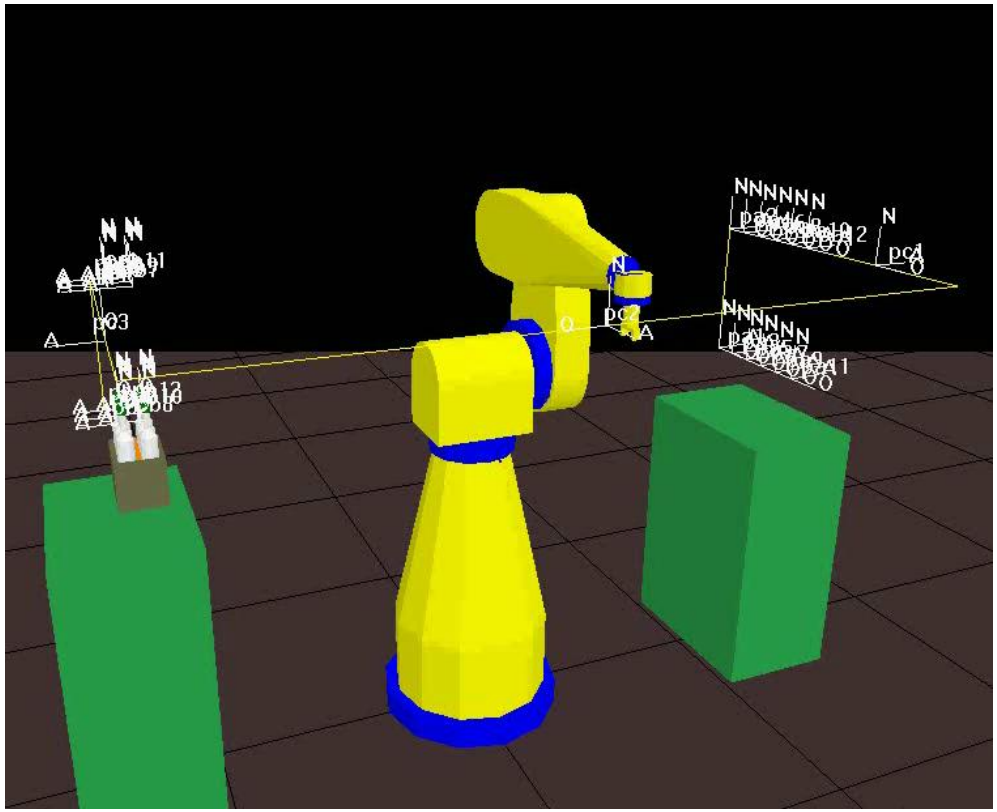
- Were is what?
- Where is who?

How to represent episodes?

- Use of Paragraphs
- Use of Lengthy Texts

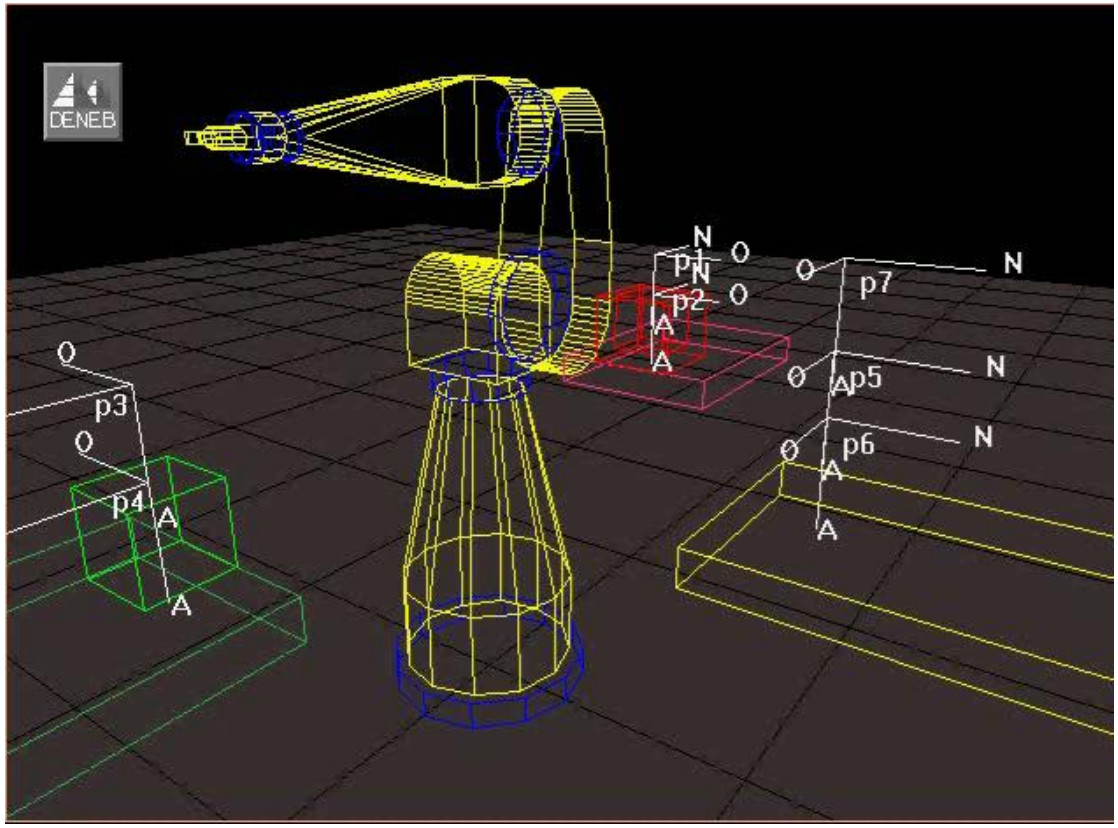
Example 1

- A robot arm picks up a bottle at incoming conveyer, and places it into a specific location inside the box at outgoing conveyor. It repeats the operations until the box is full.



Example 2

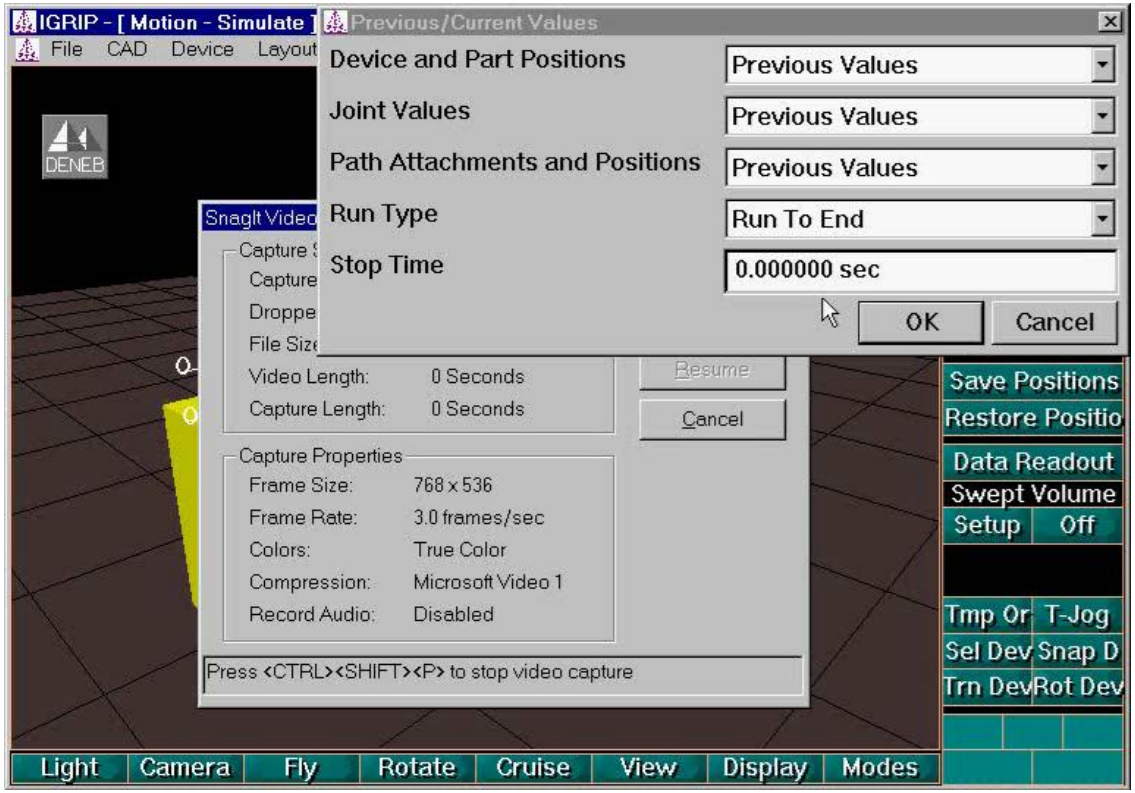
- An AGV transports part A into the workspace. A conveyor transports part B into the workspace. A robot arm picks up part A and places it onto the outgoing conveyor. The robot arm picks up part B and inserts it into part B.



Example 3

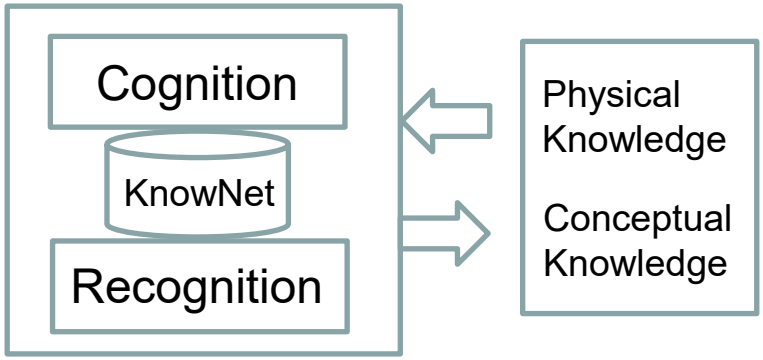
A robot arm is performing the following actions:

- To pick up a bar of one colour
- To place it onto the table of the same colour
- Repeat for the rest of three bars
- To pick up the bar on a table
- To place it to the middle of the table
- Repeat for the rest of three bars



Outline of Lecture 1

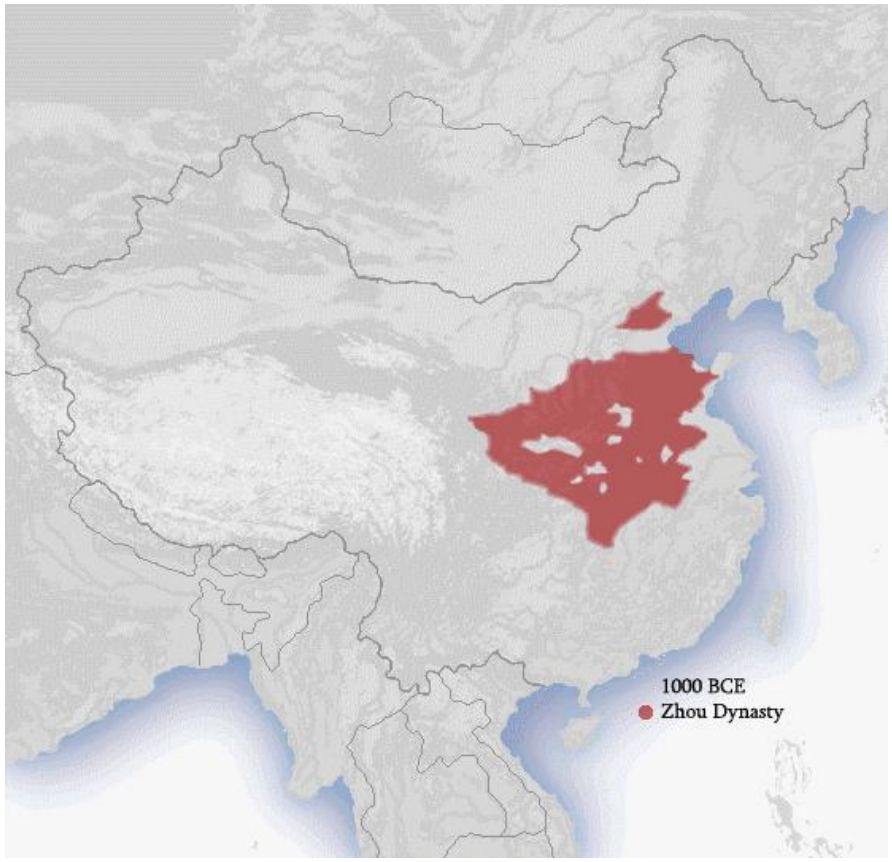
- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



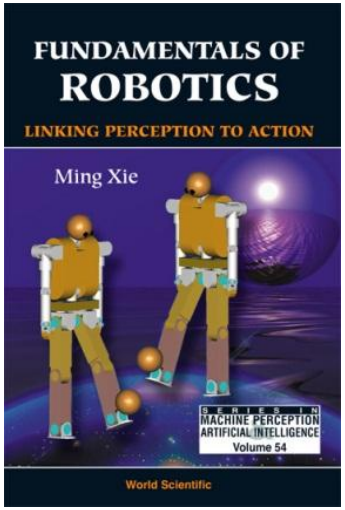
- Were is what?
- Where is who?

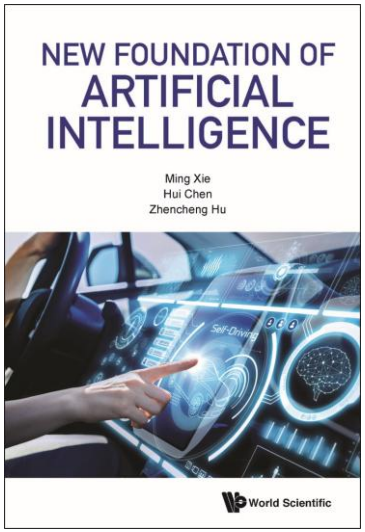
How to represent stories?

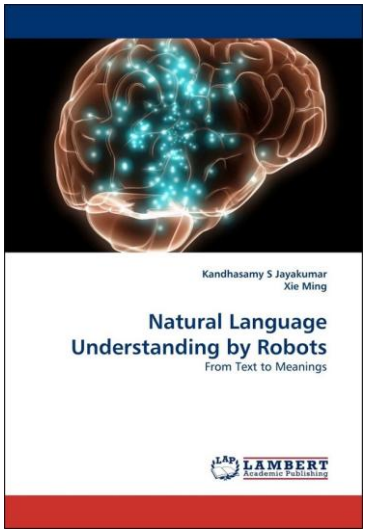
- Use of Texts Organized into Books
- Use of Texts Organized into Reports
- Use of Texts Organized into Articles

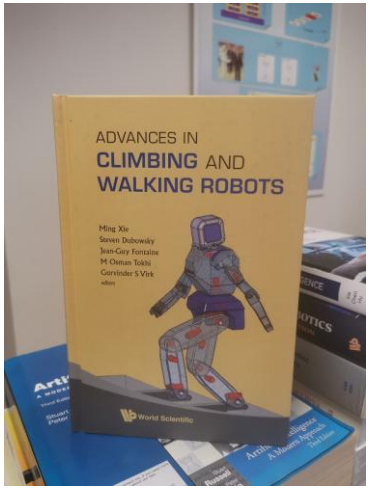


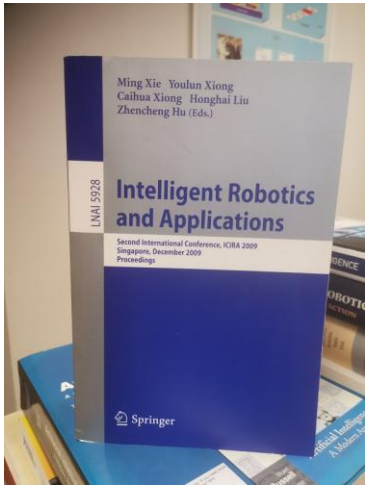
What I have done …

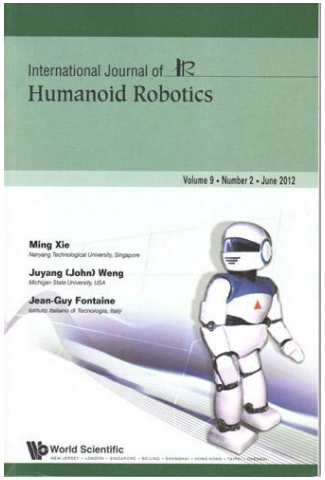






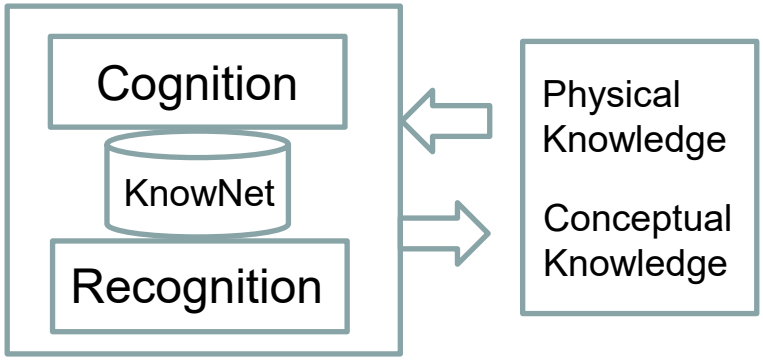






Summary of Lecture 1

- Invention of Natural Language
- Representation of Entities
- Representation of Systems
- Representation of Interactions
- Representation of Events
- Representation of Episodes
- Representation of Stories



- Were is what?
- Where is who?

Outline of Module 2

- Lecture 1:
  - Use of Natural Language
- Lecture 2:
  - Use of Technical Language
- Lecture 3:
  - Use of Programming Language

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 2 of Module 2

AI 3.0

MA4829 Machine Intelligence

Use of Technical Language to Represent Knowledge

XIE Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

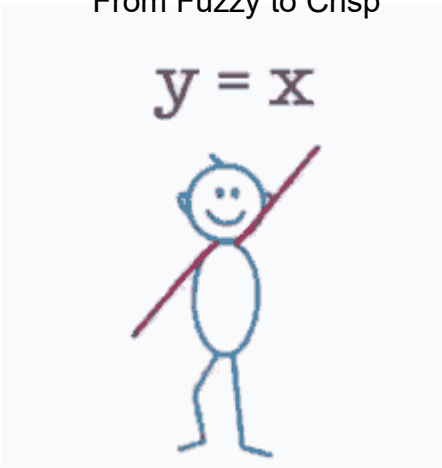
"We are living inside an ocean of signals"

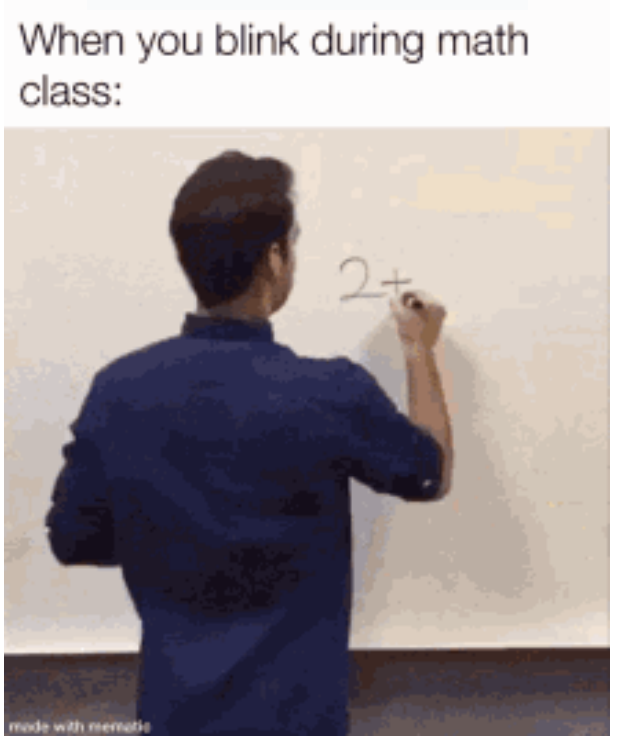
What is a system with intelligence inside?

Outline of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

From Fuzzy to Crisp

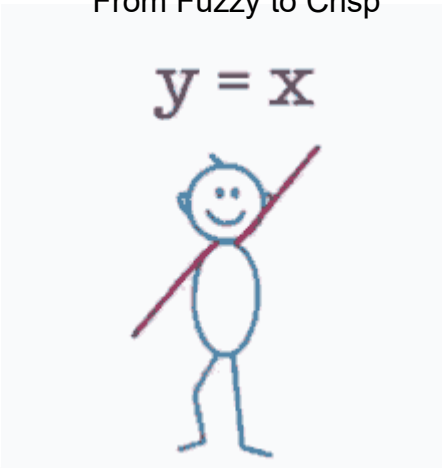


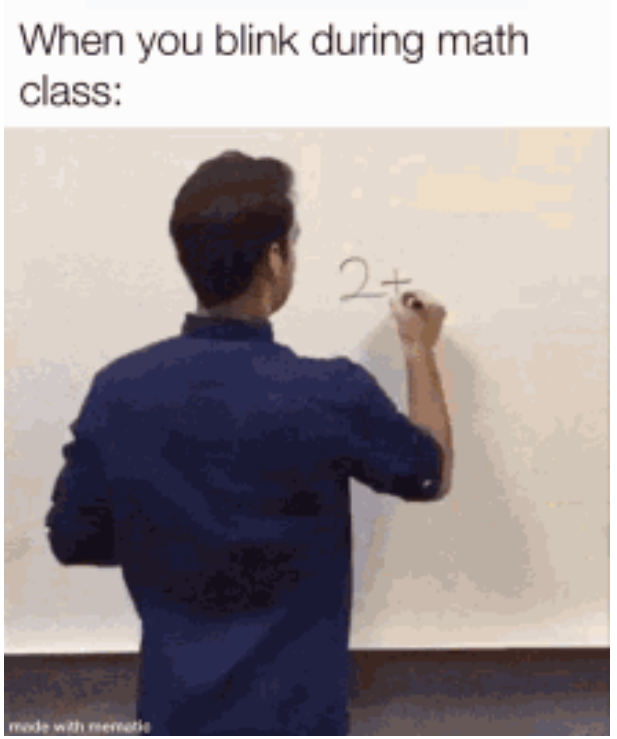


Outline of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

From Fuzzy to Crisp





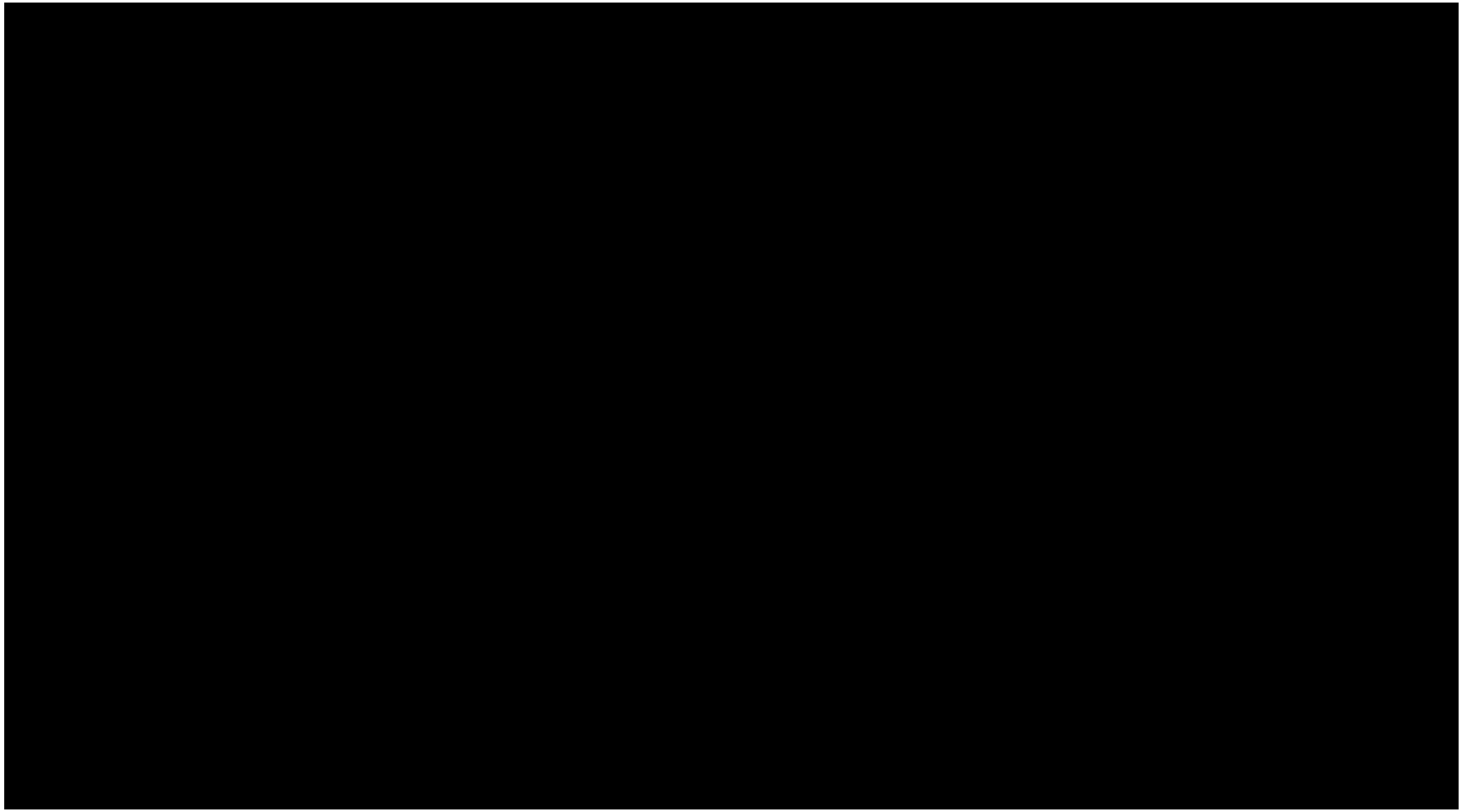
What are technical languages?

Answer:

- They are the extension of natural languages
- They are in the forms of symbols and equations, which describe the properties and constraints (i.e. knowledge) in the physical world so as to facilitate compact representations and precise computations manually and/or automatically.

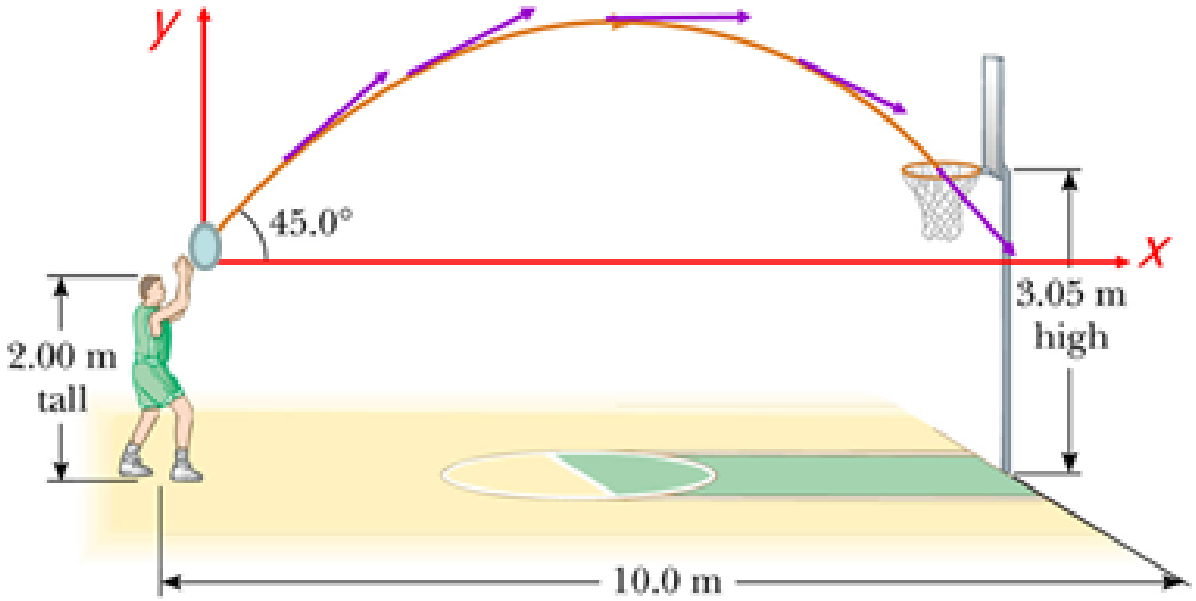
Is a technical language an invention?

How are technical languages invented?



Example of Using Technical Language

- How to scientifically provide knowledge to the player for him/her to score the shooting of a basketball in the following scenario?



Answer

- To launch the ball at the height of 2.0 meters within a vertical plane.

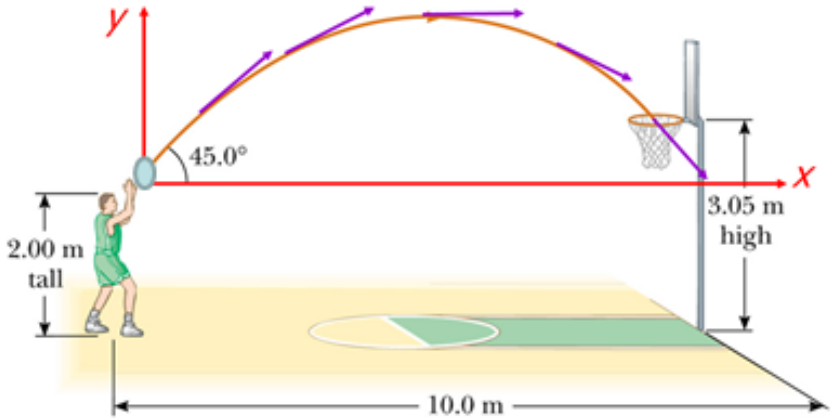
- To launch the ball at the angle of 45.0 degrees with the initial speed of v0, which is calculated by using the following two equations:

$$\Delta y = 3.05 - 2.0 = v_0 \sin(45^0)t + \frac{1}{2}(-g)t^2$$

$$\Delta x = 10.0 = v_0 \cos(45^0)t$$

To apply force F0 for 0.2 seconds to generate v0. F0 is calculated from:

$$F_0 = \frac{mv_0}{0.2}$$

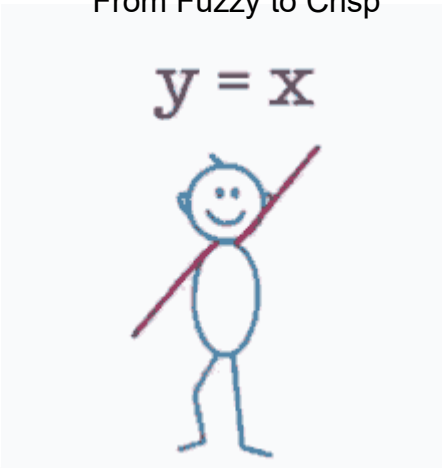


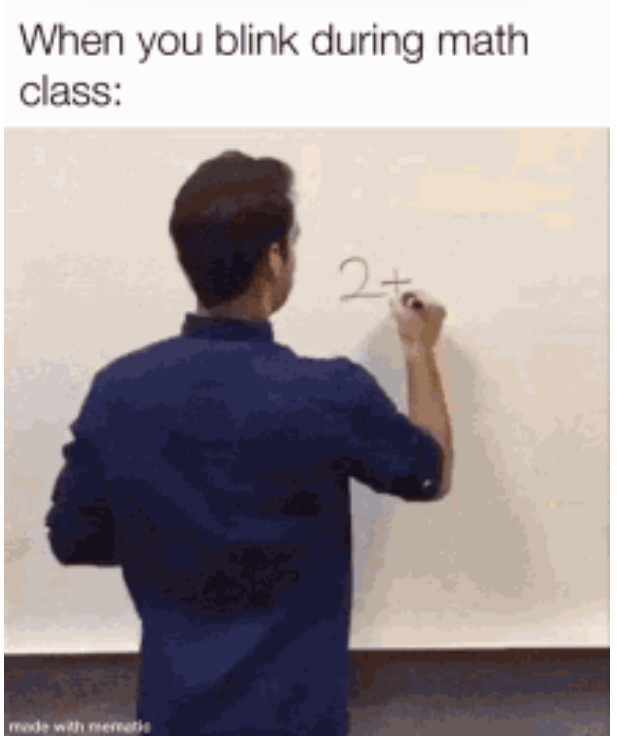
A Demonstration …

Outline of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

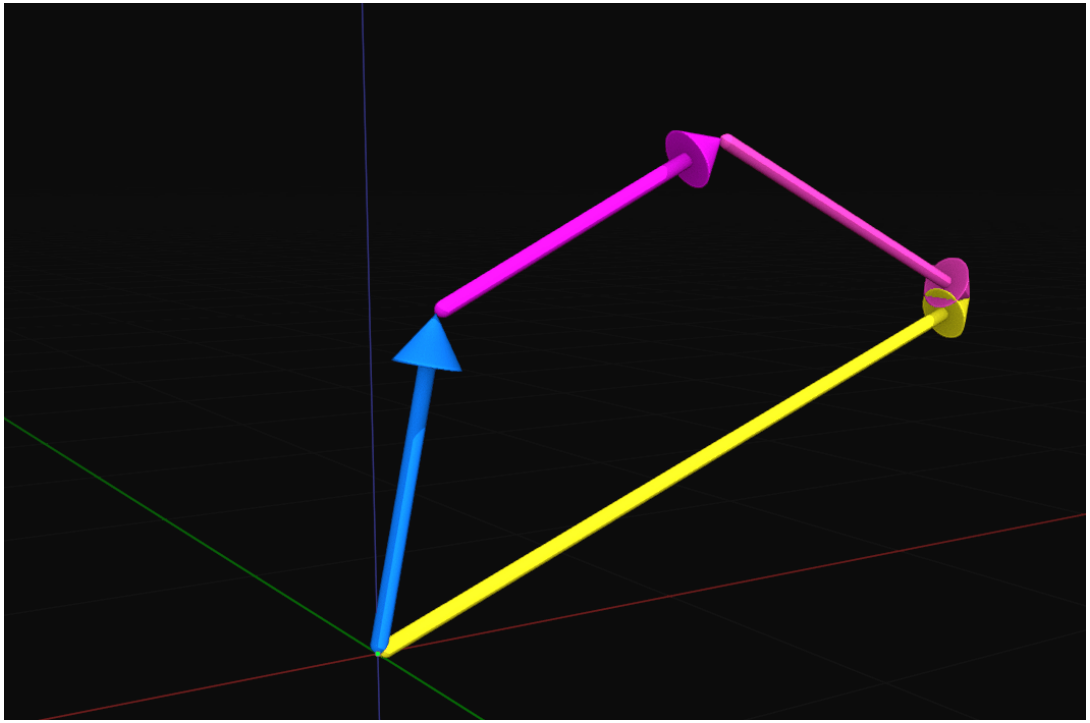
From Fuzzy to Crisp



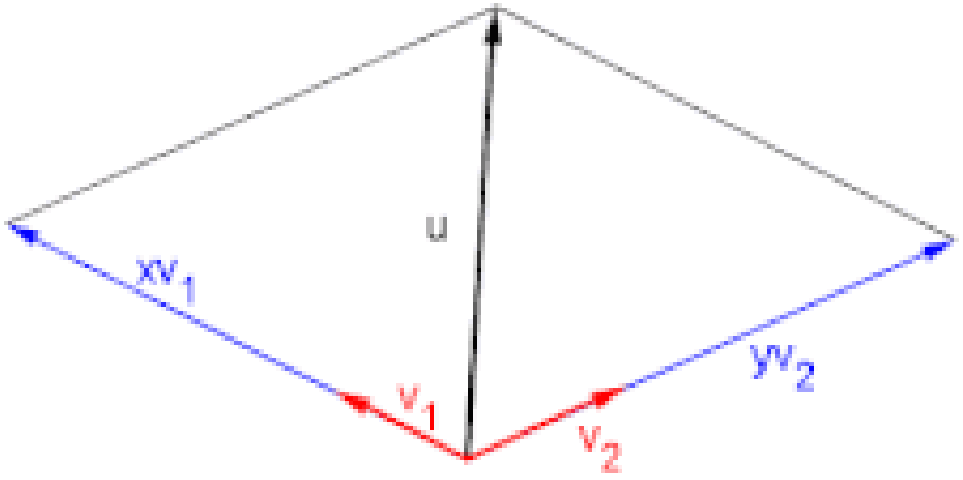


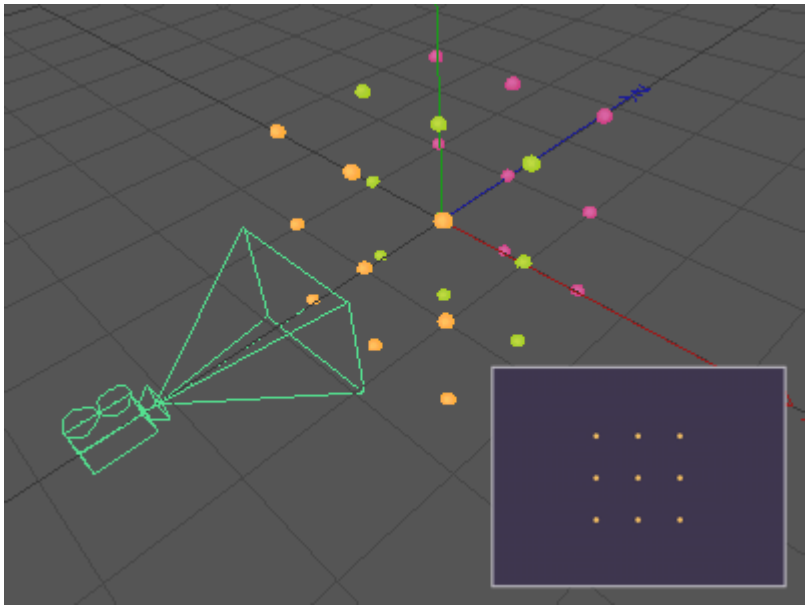
What is a vector?

- It is a list of one-dimensional elements, each of which represents a meaningful value.
- A vector has a head
- A vector has a tail
- A vector has a length
- A vector has a phase

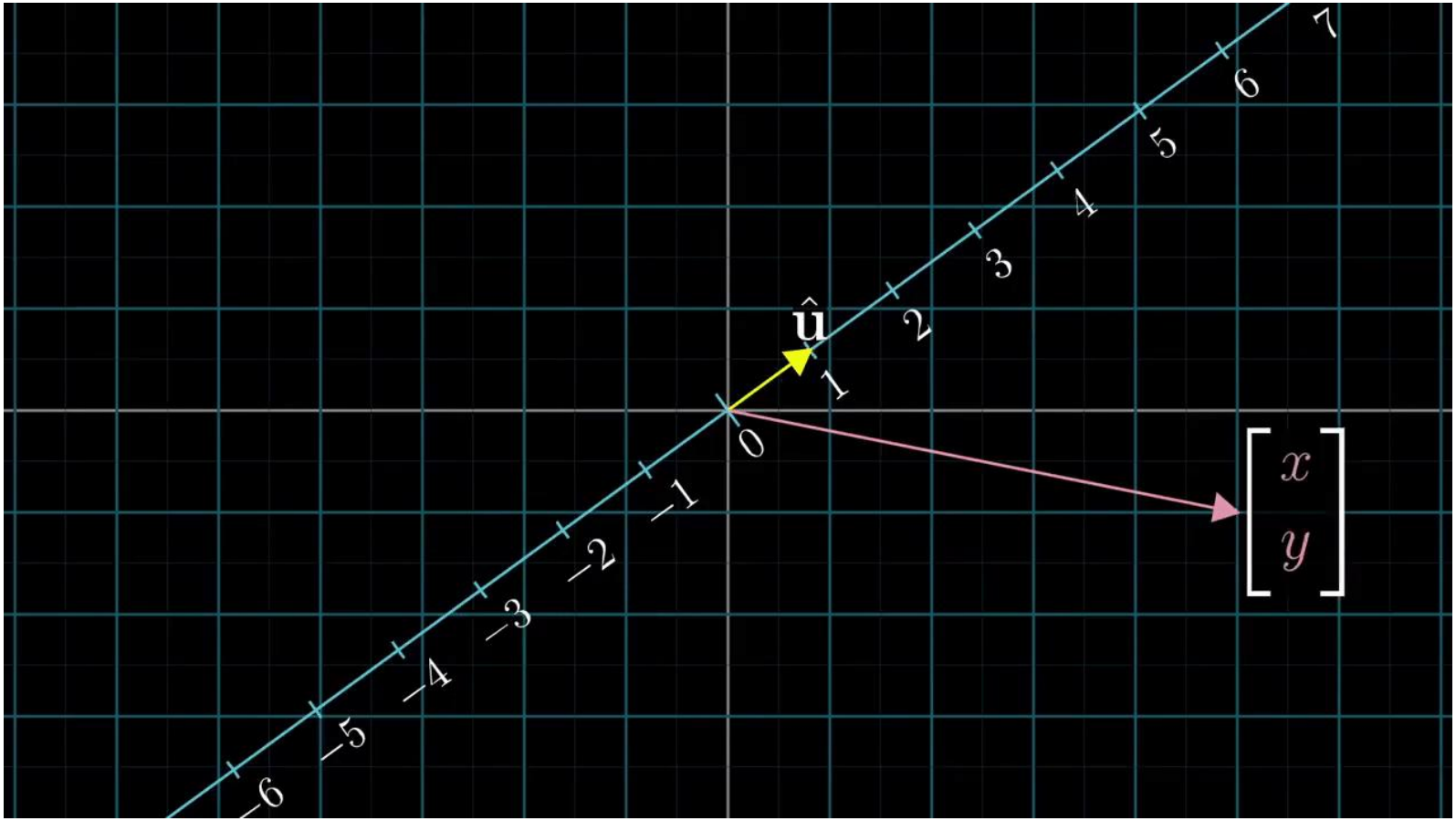


Example of Vectors in 2D Space

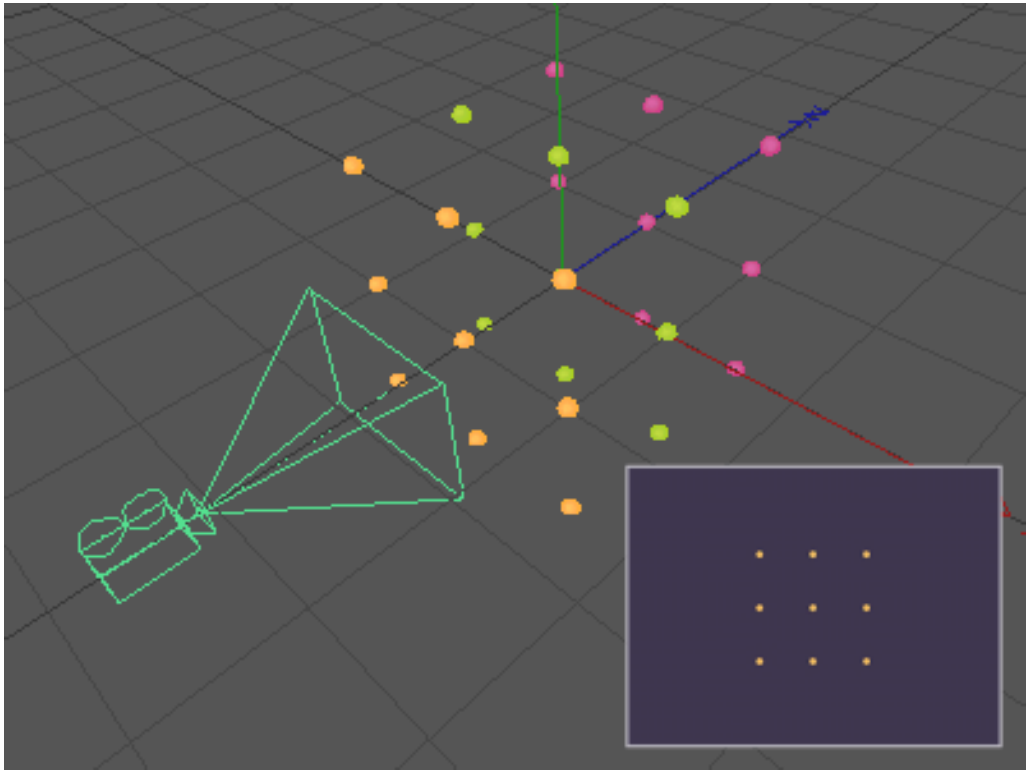


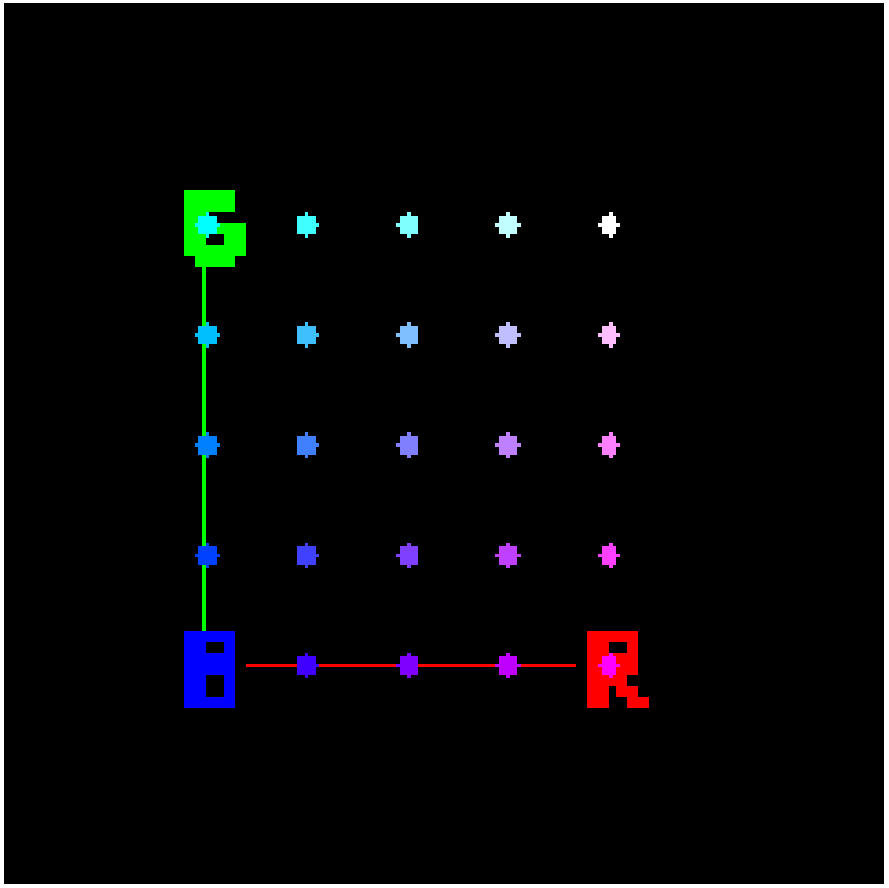


Transformation of Vectors in 2D Space

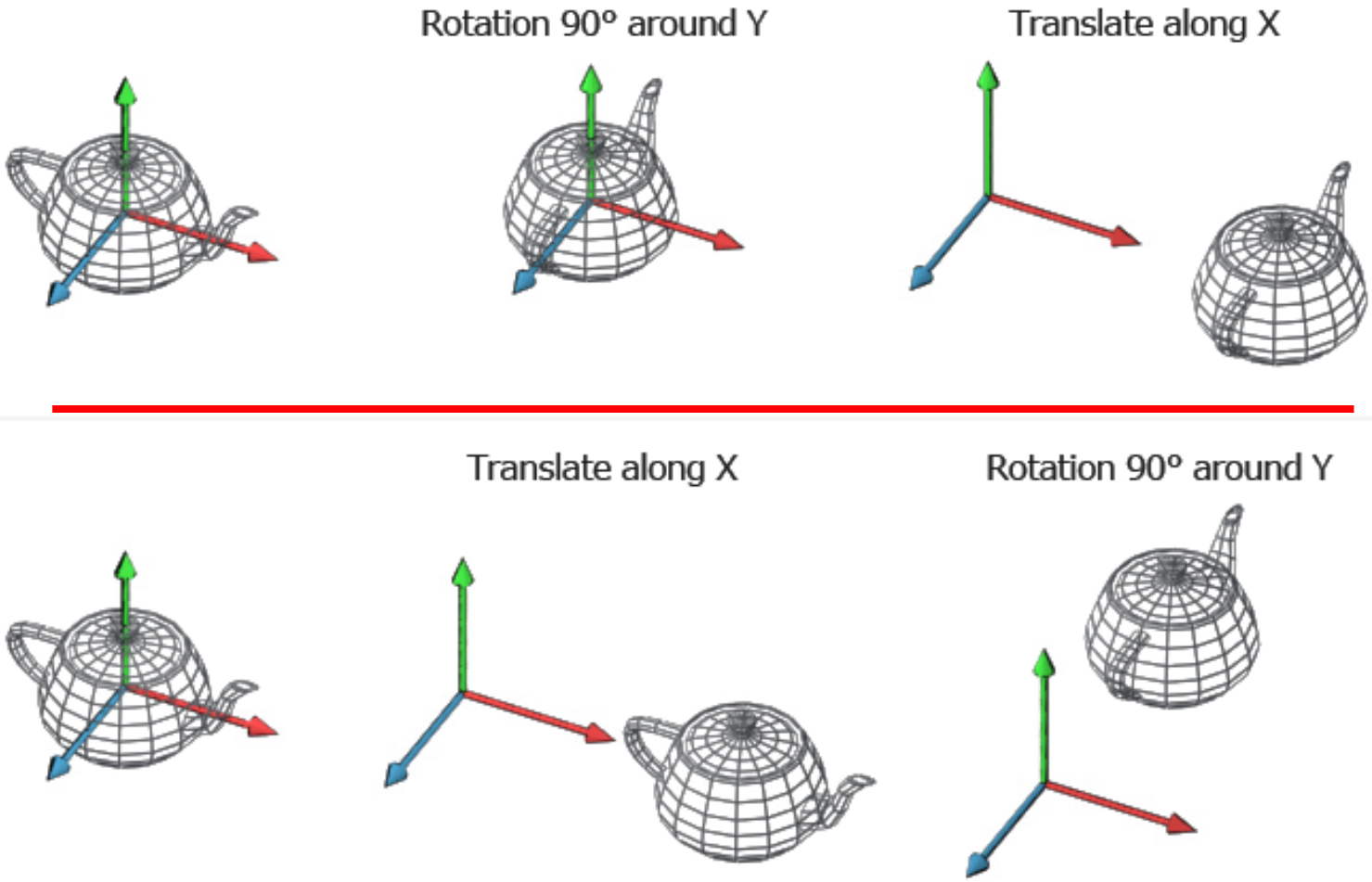


Example of Vectors in 3D Space





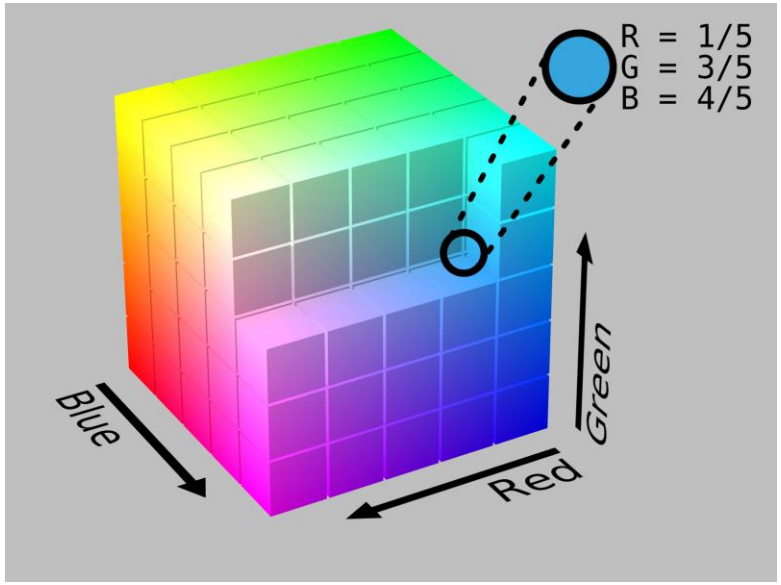
Transformation of Vectors in 3D Space



Example of Feature Vector or Knowledge Vector of Higher Dimensions …

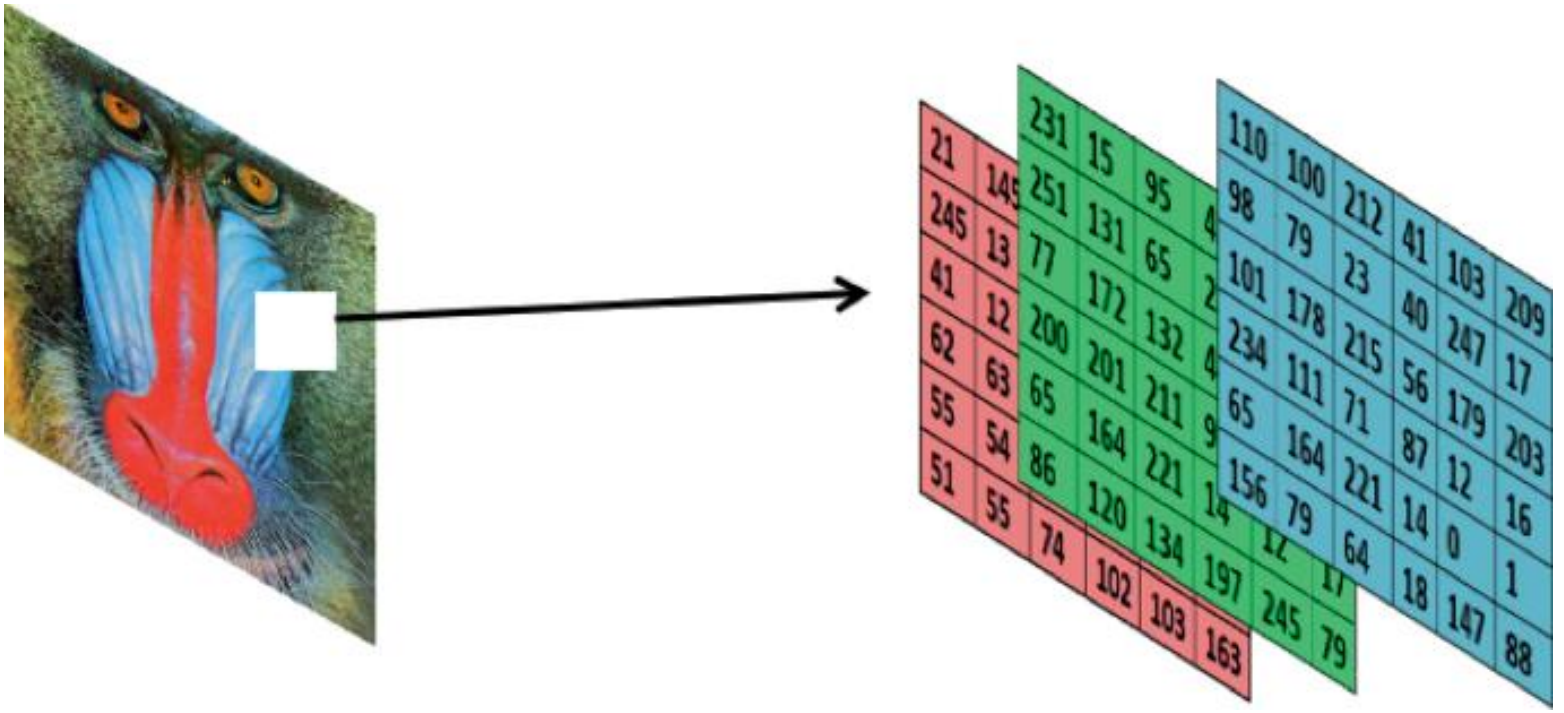
$$V_k = \{(f_1, f_2, \ldots, f_i, \ldots f_n), i = 1,2, \ldots, n\}$$

f_i: i-th feature or meaningful value



Example of RGB Color Vectors

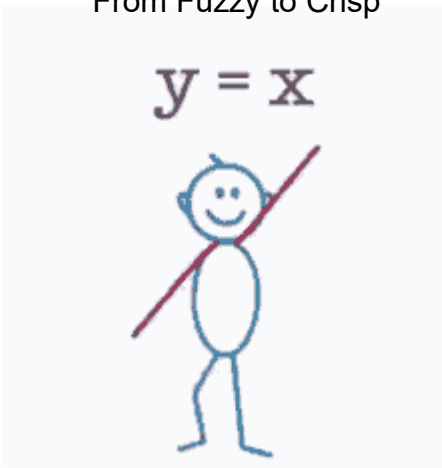
- At location (0, 0), red = 21, green = 231, blue = 110

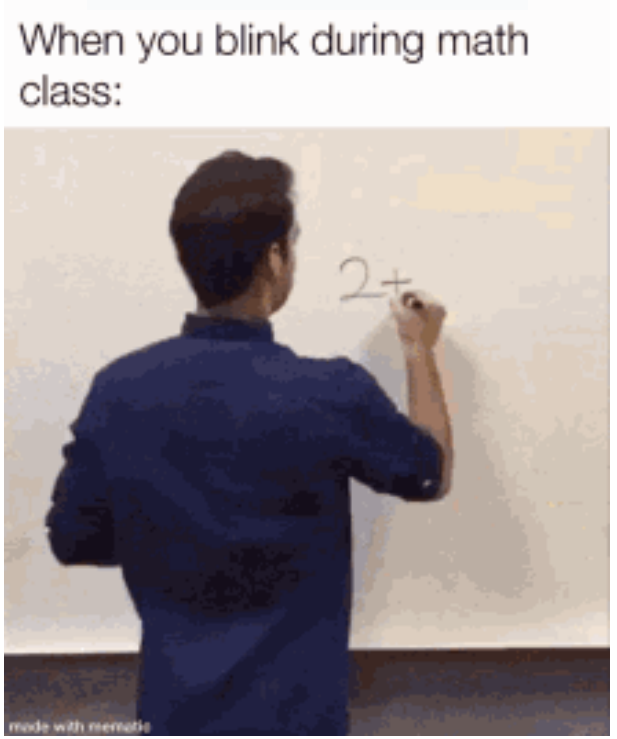


Outline of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

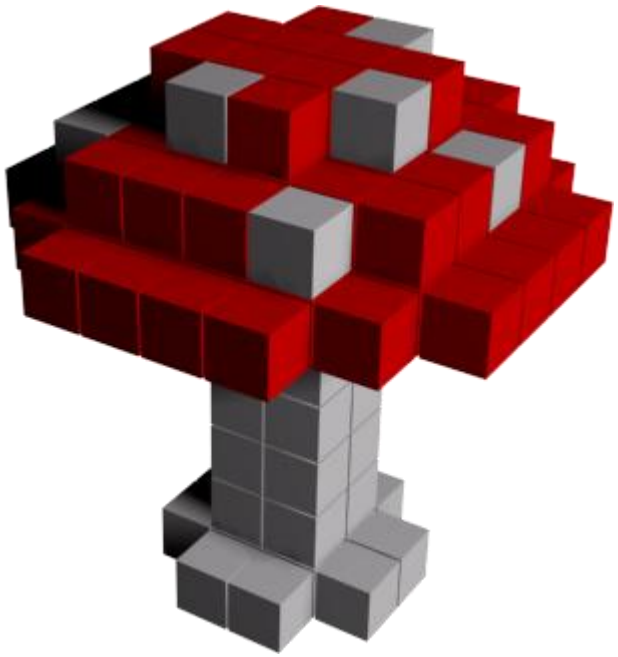
From Fuzzy to Crisp





What is a matrix?

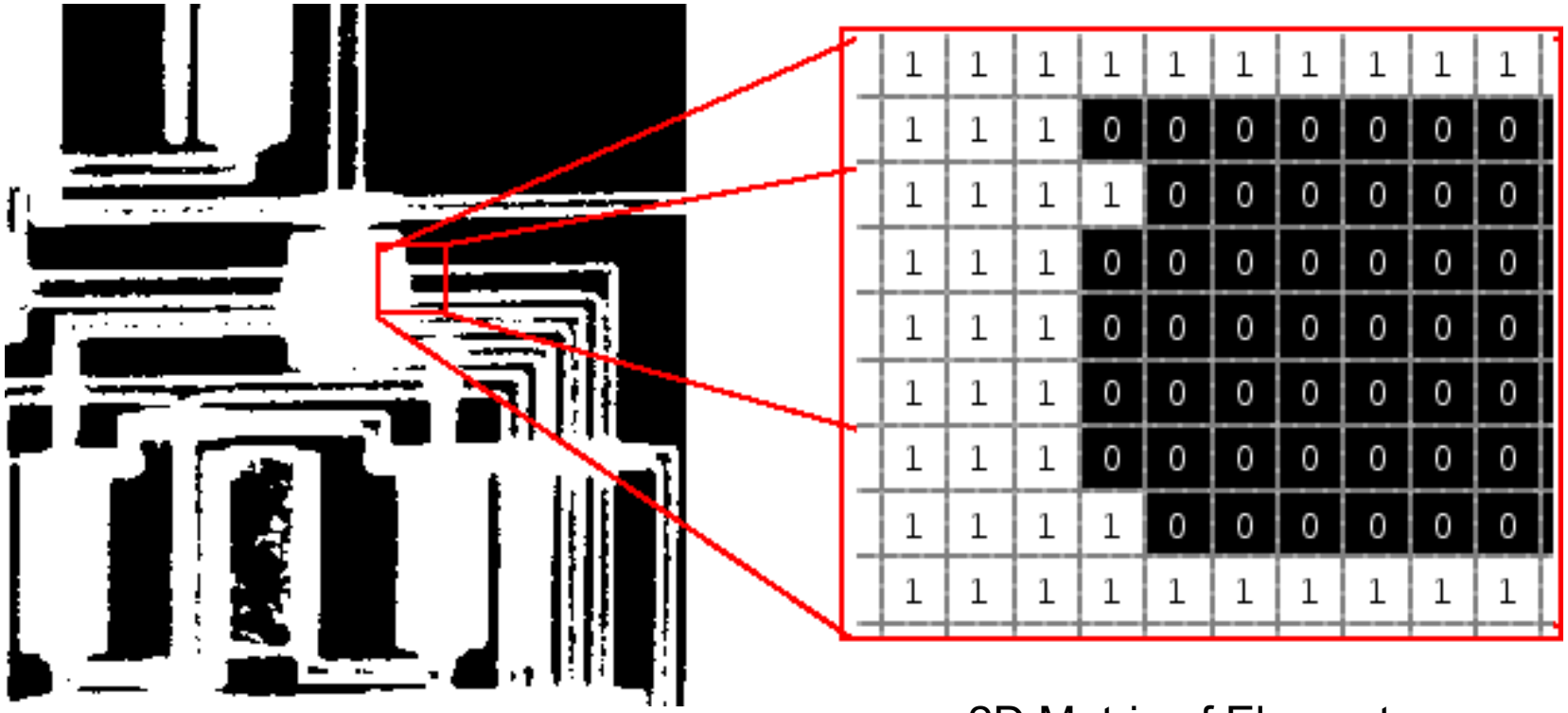
- If we consider a vector is a one-dimensional matrix, a matrix of dimension N is a list of vectors of dimension N-1.



3D Matrix of Voxels

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Example of 2D Image Representation



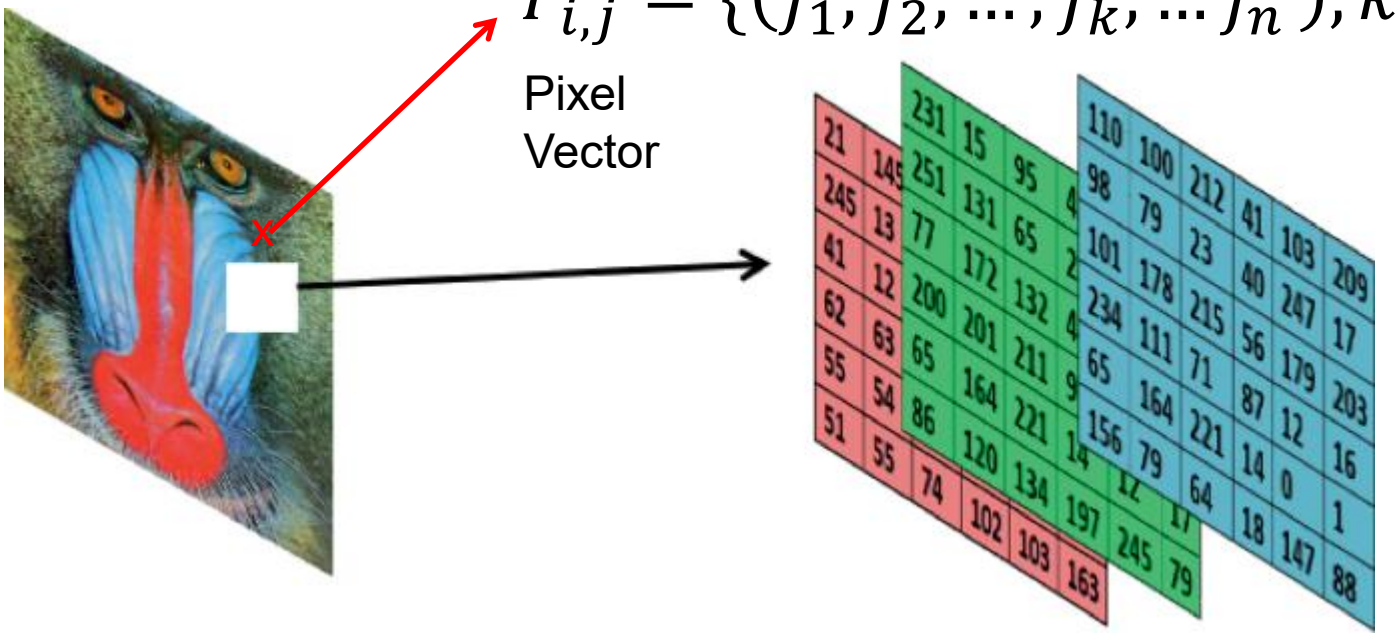
2D Matrix of Elements

In general, an image is represented as a 2D matrix of meaningful pixel vectors …

$$I = \{P_{i,j}, i = 0,1,2, \dots ; j = 0,1,2, \dots \}$$

$$P_{i,j} = \{(f_1, f_2, \dots, f_k, \dots f_n), k = 1,2, \dots, n\}$$

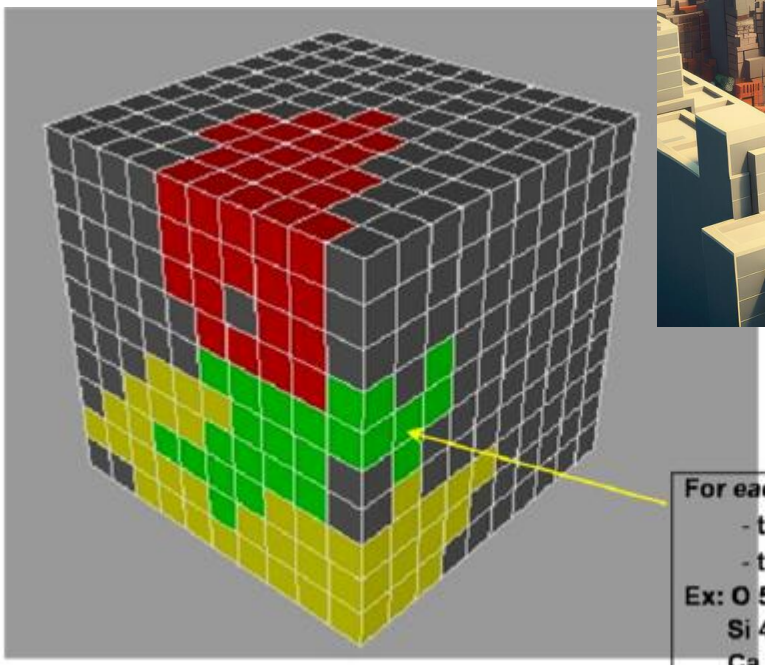
Pixel Vector



2D Matrix of Pixel Vectors

Example of 3D Scene Representation

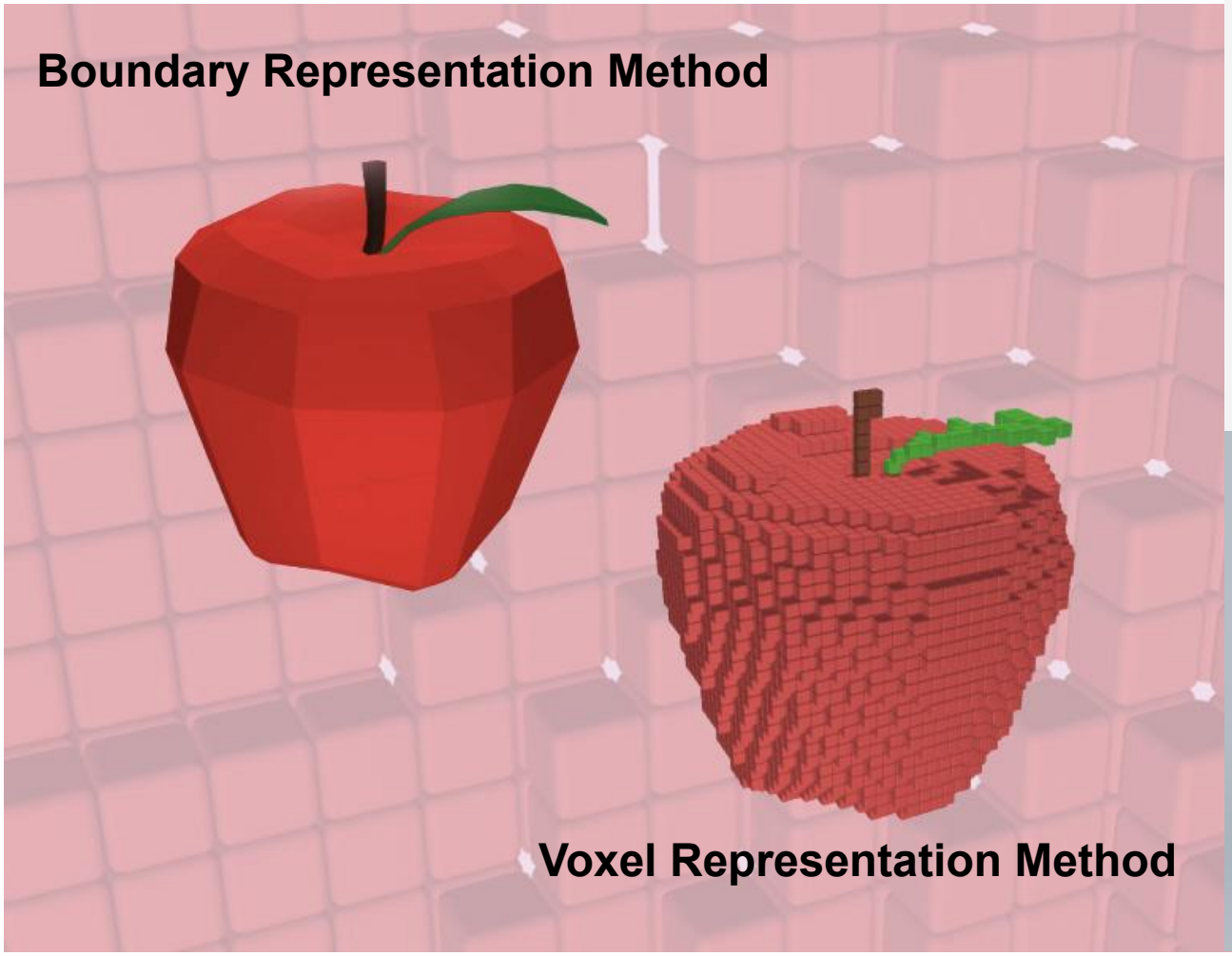
3D Matrix of Vectors

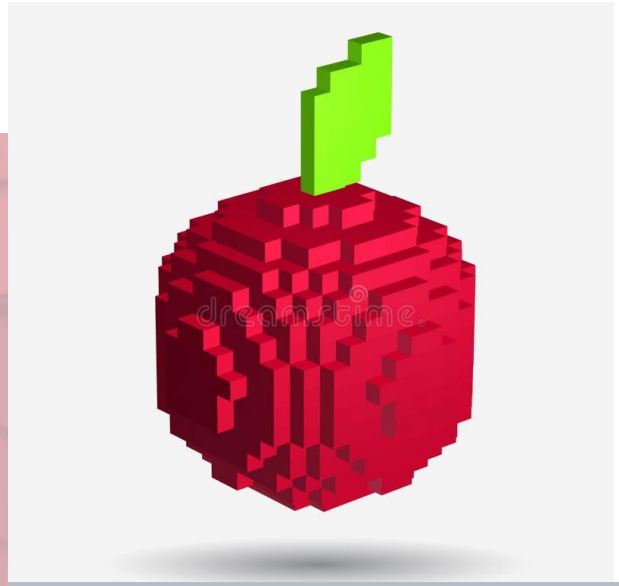


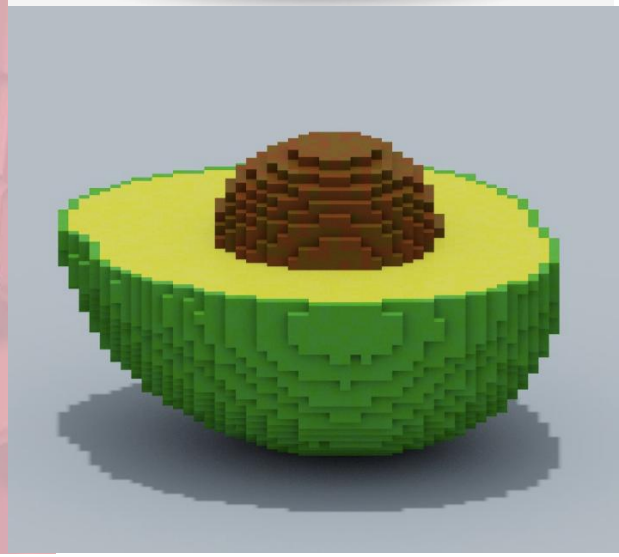
For *each grid voxel* is specified:
- the chemical composition
- the density

Ex: O 50%
Si 40 %
Ca 10%
$\rho = 2.2\ g/cm^3$

More Examples ...





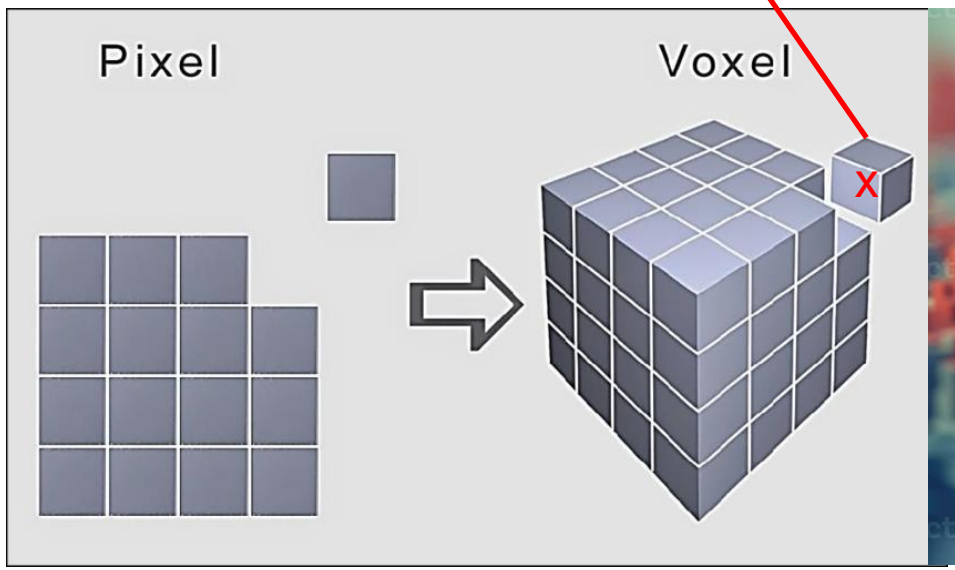


In general, a scene is represented as a 3D matrix of meaningful voxel vectors …

$$S = \{V_{i,j,k}, i = 0,1,2, \dots; j = 0,1,2, \dots; k = 0,2, \dots\}$$

Voxel Vector

$$V_{i,j,k} = \{(v_1, v_2, \dots, v_n, \dots, v_m), n = 1,2, \dots, m\}$$

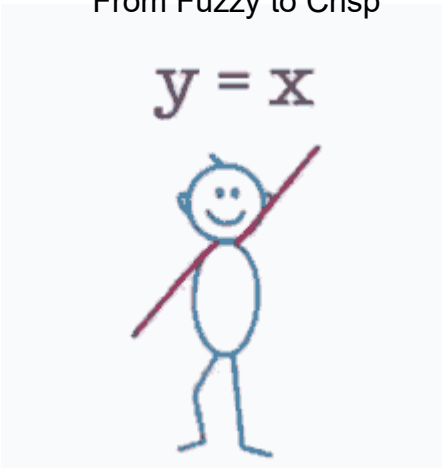


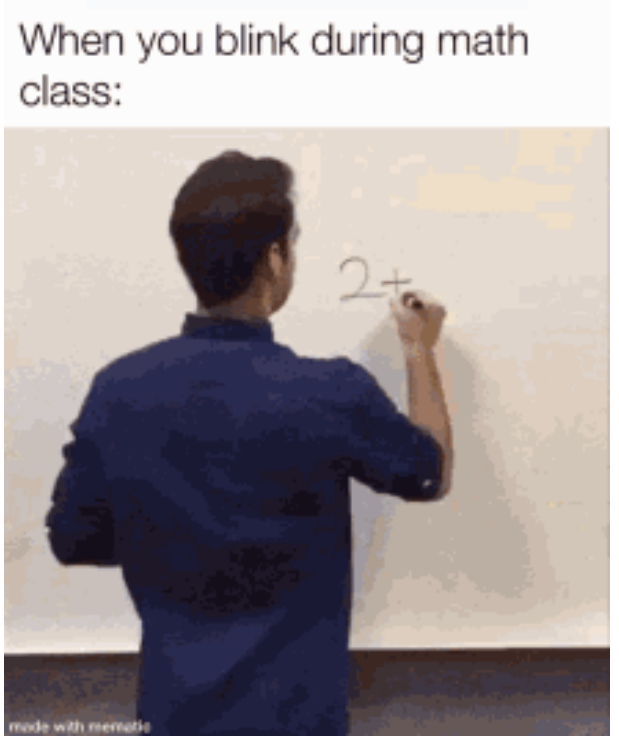
3D Matrix of Voxel Vectors

Outline of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

From Fuzzy to Crisp



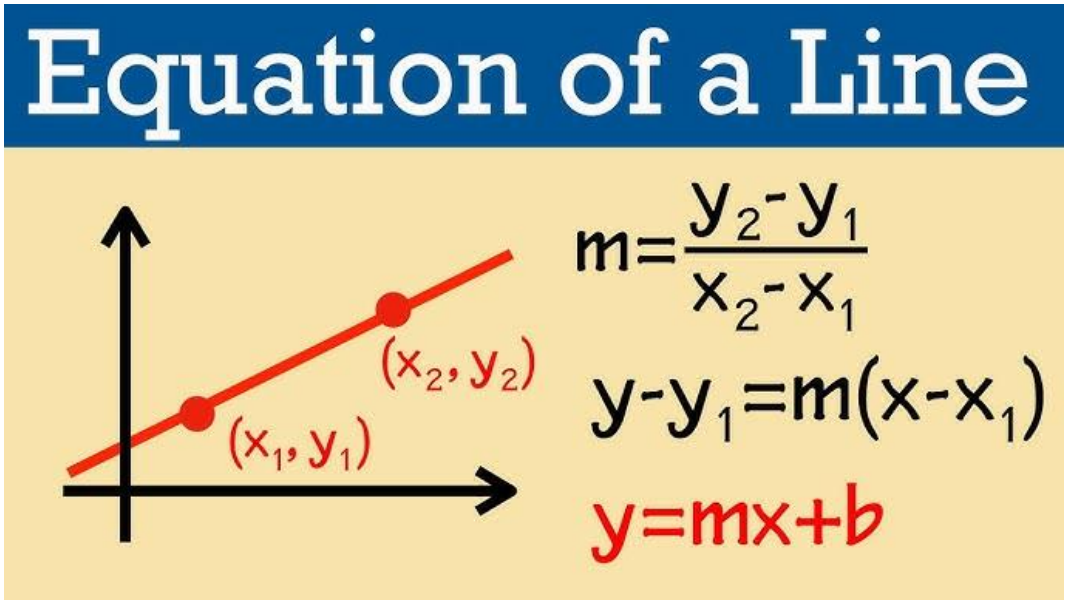


What is a straight line?

- It refers to a set of consecutive points in which the gradients of coordinates are constant.
- A line does not change direction.
- A line does not have curvature.
- A line's thickness is constant.

Equation of Lines in 2D Space

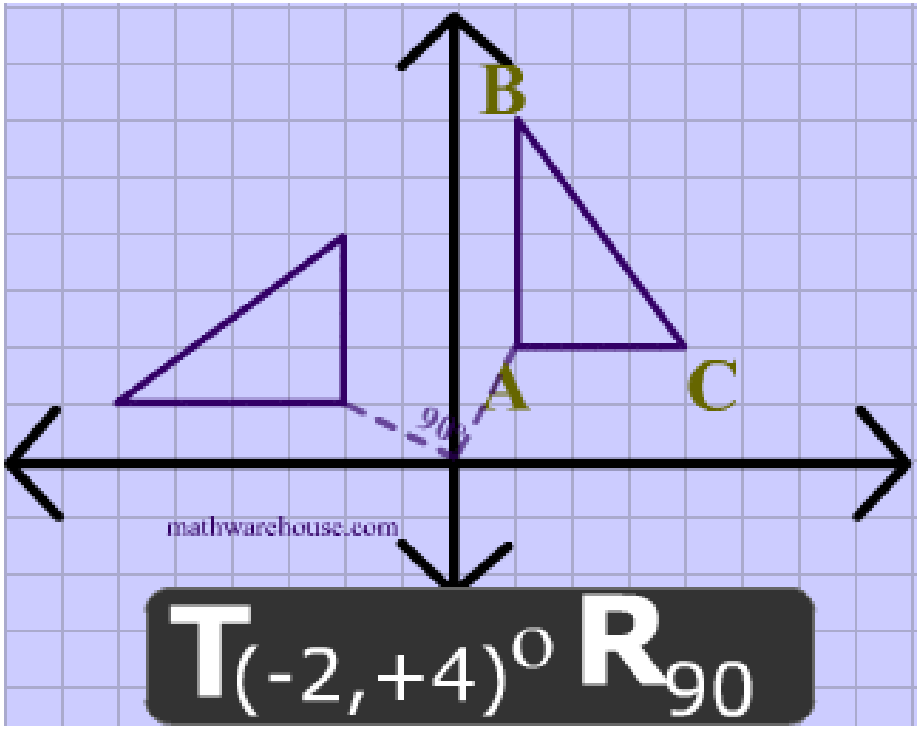
$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1} \Rightarrow ax + by + c = 0$$



Transformation of Lines in 2D space

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = R_{2x2} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = T_{2x1} + \begin{bmatrix} x \\ y \end{bmatrix}$$

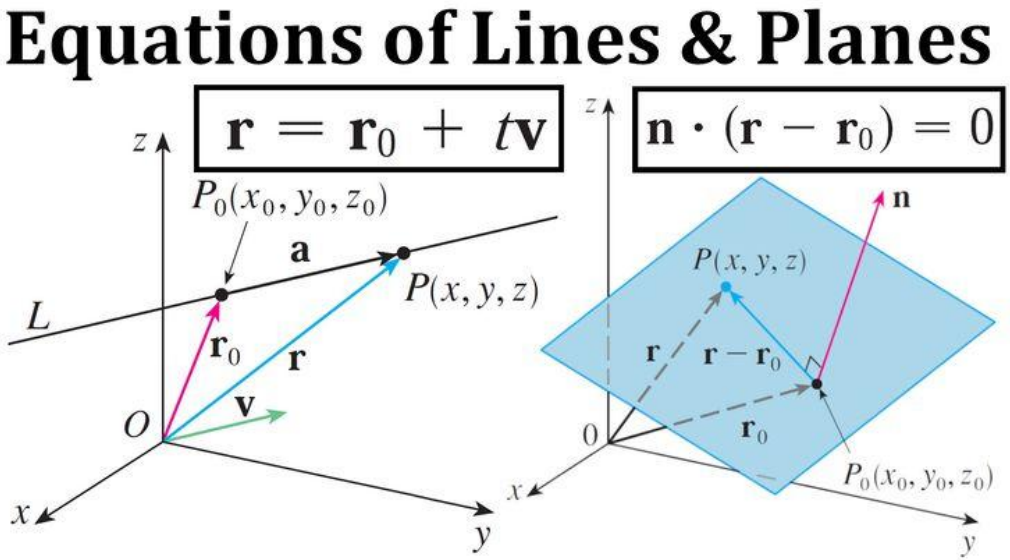


Equation of Lines in 3D Space

$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1} \quad \text{and} \quad \frac{x - x_1}{z - z_1} = \frac{x_2 - x_1}{z_2 - z_1}$$

or:

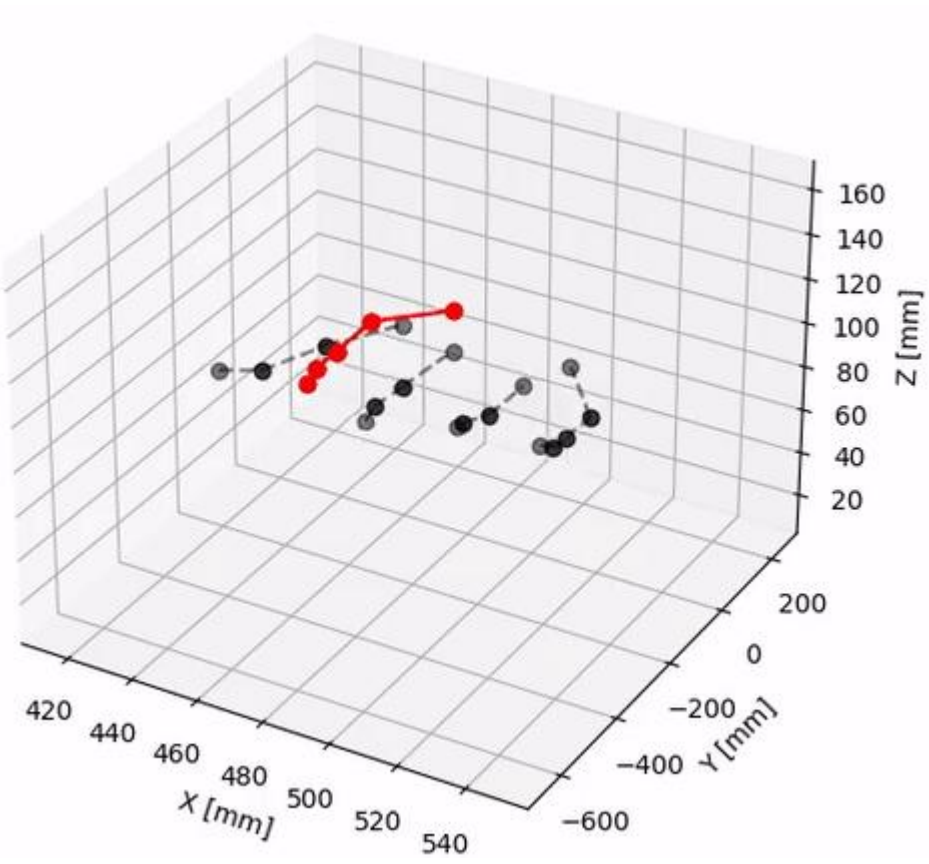
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + k \cdot \vec{v}$$



Transformation of Lines in 3D Space

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = R_{3x3} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

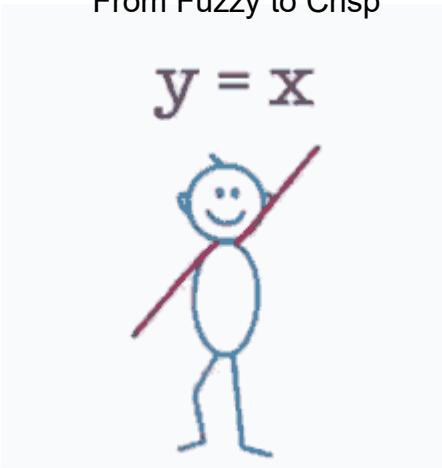
$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = T_{3x1} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

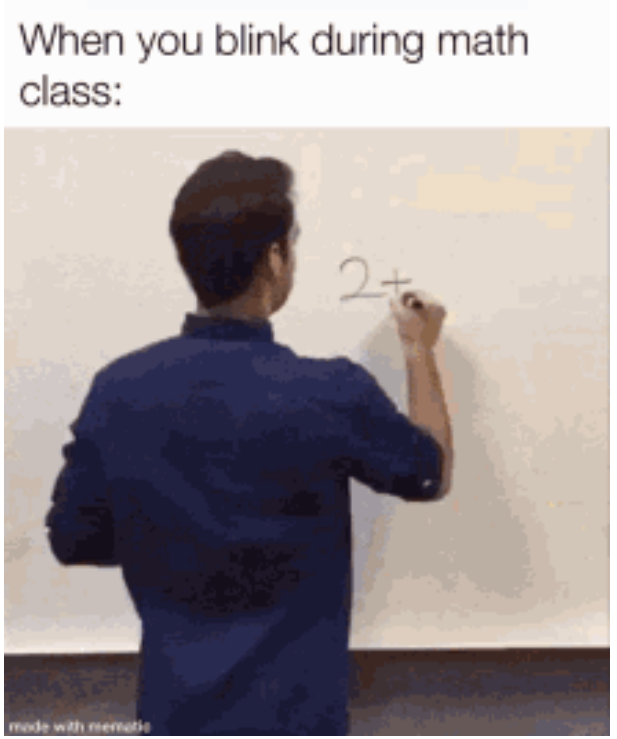


Outline of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

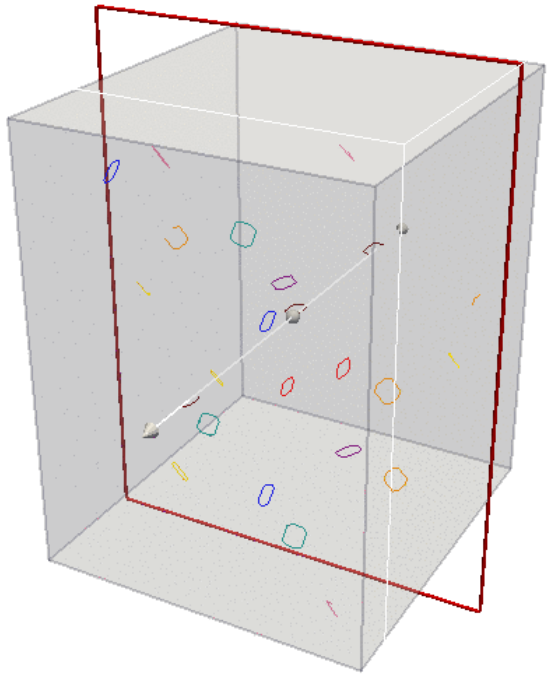
From Fuzzy to Crisp



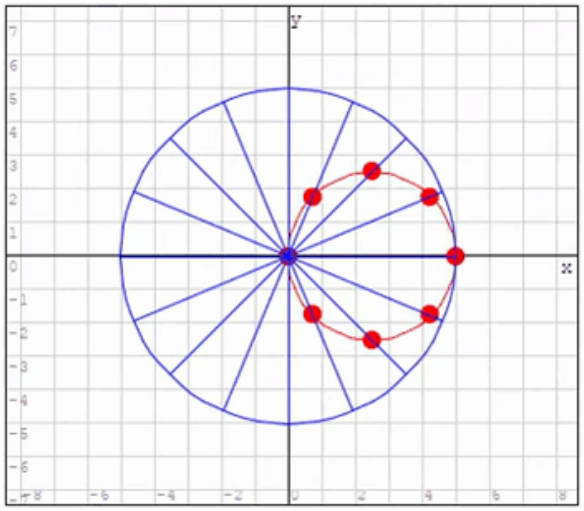


What is a curve?

- It refers to a set of consecutive points in which the gradients of coordinates are not constant.
- A curve does not have a constant direction.
- A curve's thickness is constant.

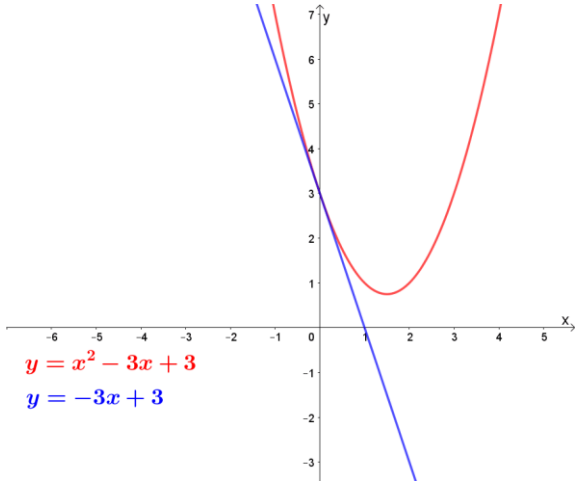


Equations of Curves in 2D Space



- Equation of Circle:

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

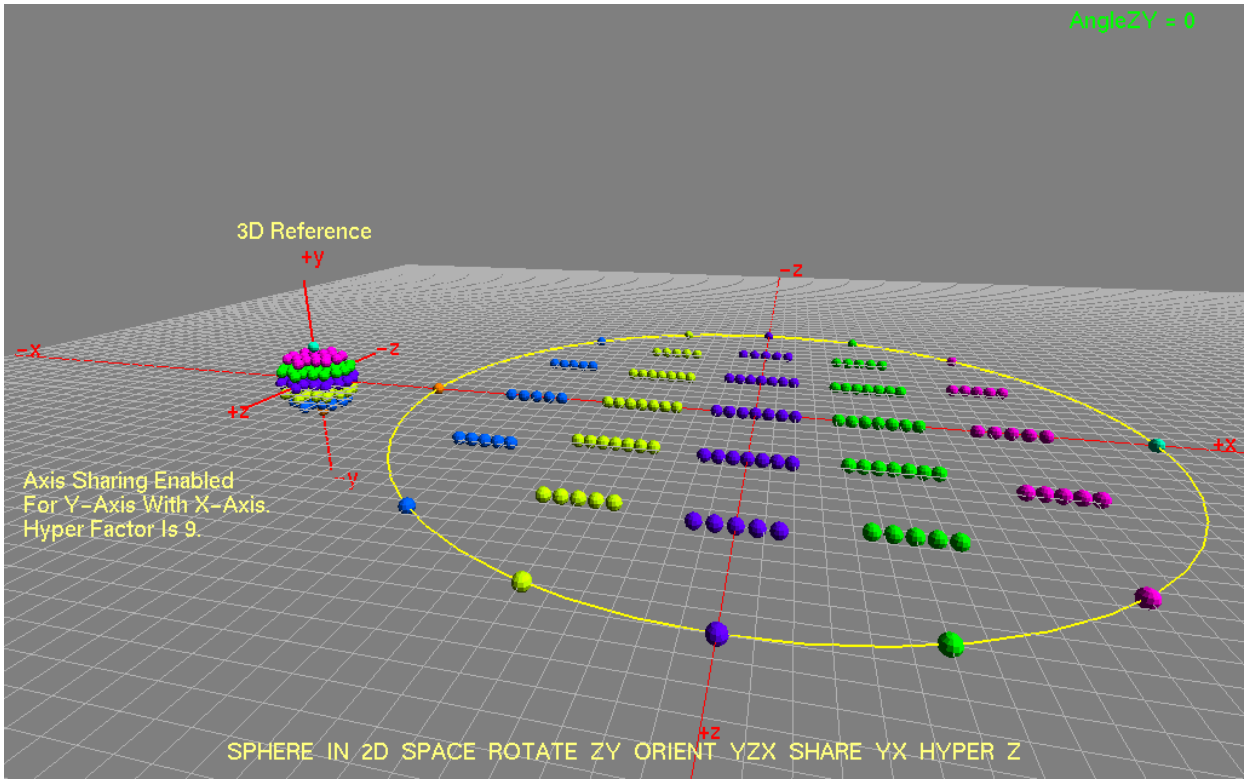


- Equation of Parabola:

$$y = ax^2 + bx + c$$

Transformation of Curves in 2D space

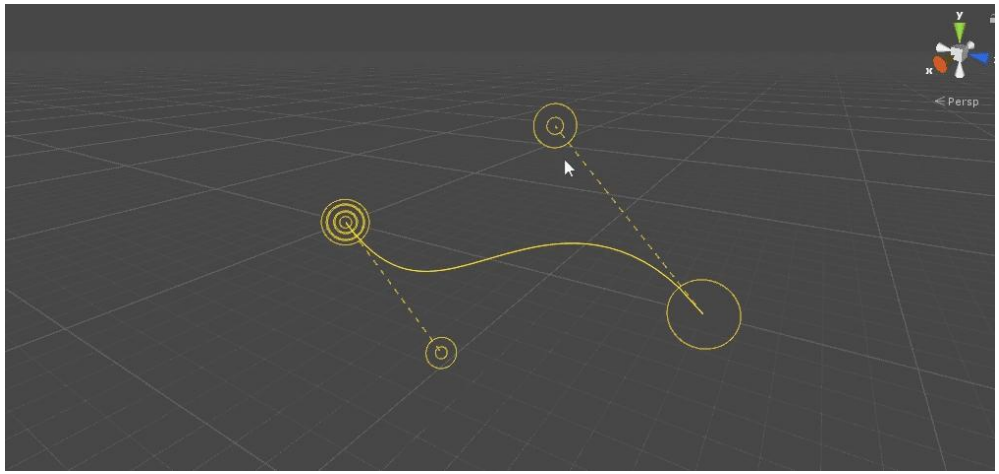
$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = R_{2x2} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = T_{2x1} + \begin{bmatrix} x \\ y \end{bmatrix}$$



Equations of Curves in 3D Space

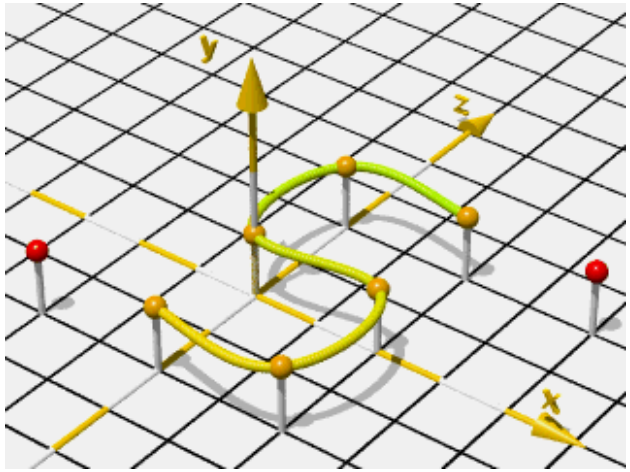
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \cdot t + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \cdot t^2 + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \cdot t + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$



$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} \cdot t^3 + \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \cdot t^2 + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \cdot t + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} \cdot t^4 + \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} \cdot t^3 + \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \cdot t^2 + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \cdot t + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

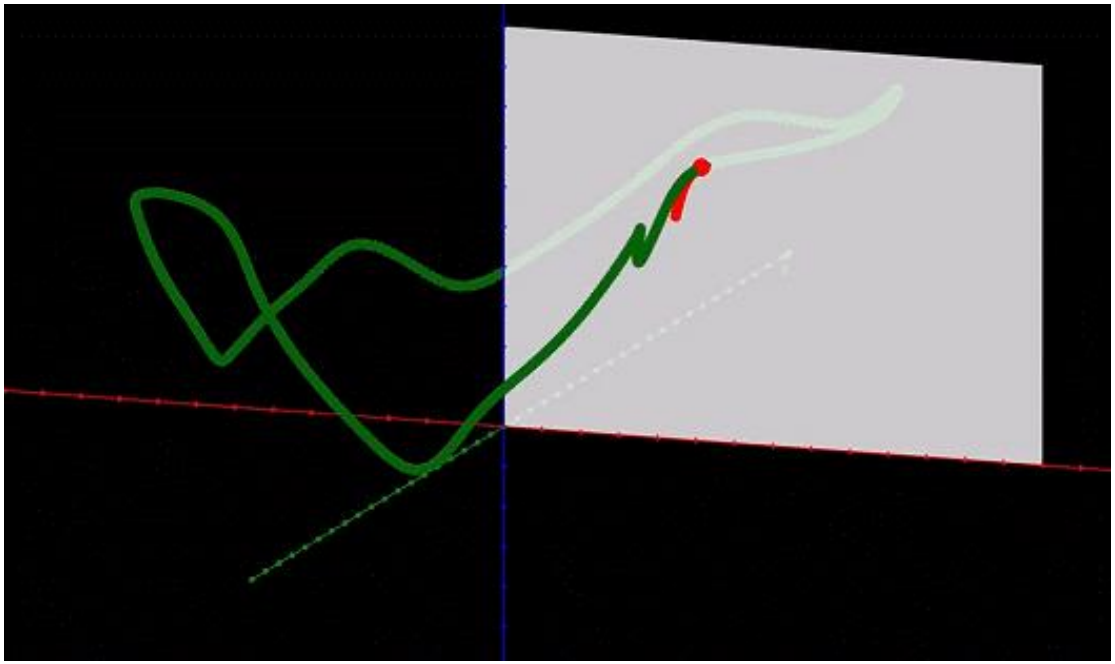


so on …

Transformation of Curves in 3D Space

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = R_{3x3} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

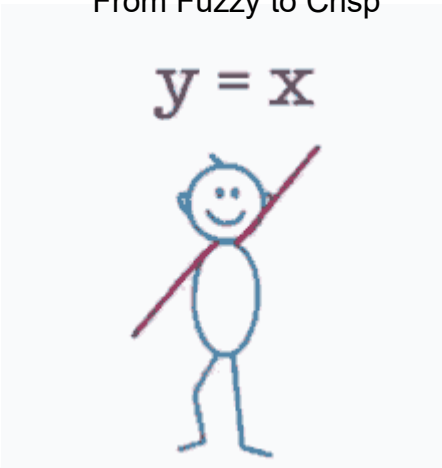
$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = T_{3x1} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

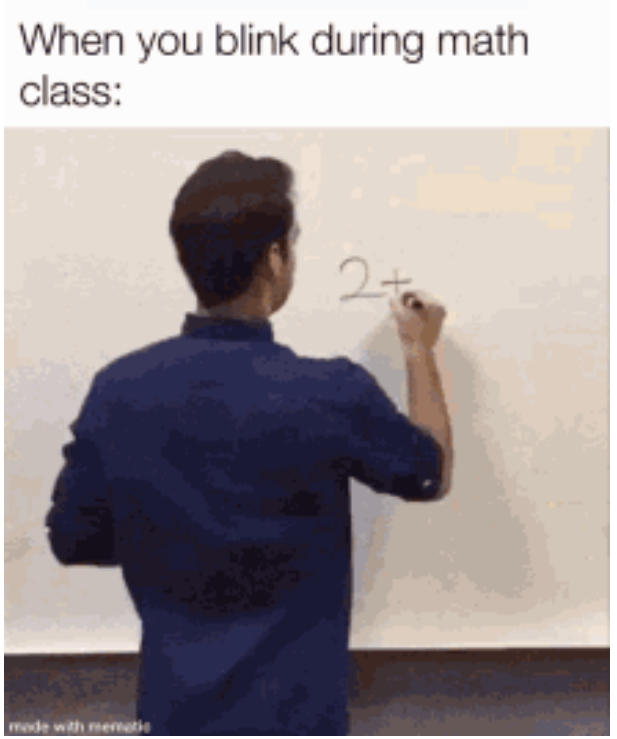


Outline of Lecture 2

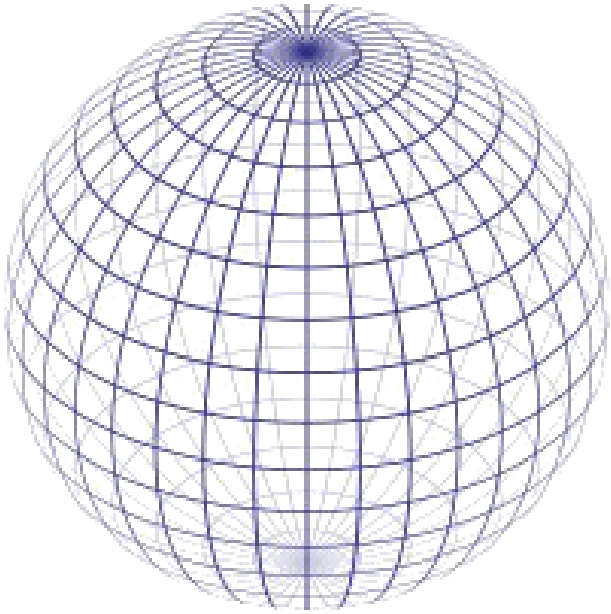
- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

From Fuzzy to Crisp

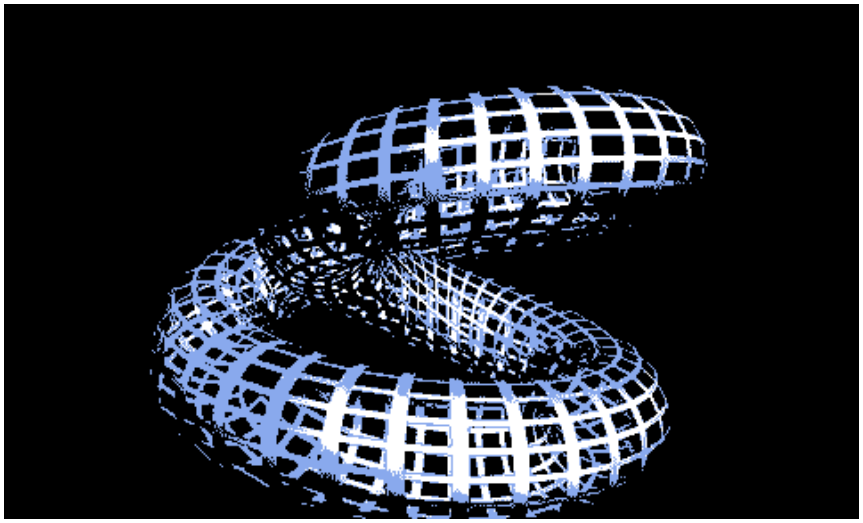




What is a surface?



- It refers to a 2D matrix of massively interconnected points.
- A surface has a normal vector at each of its points.
-
- A surface does not have volume.
- Two surfaces could intersect each other.

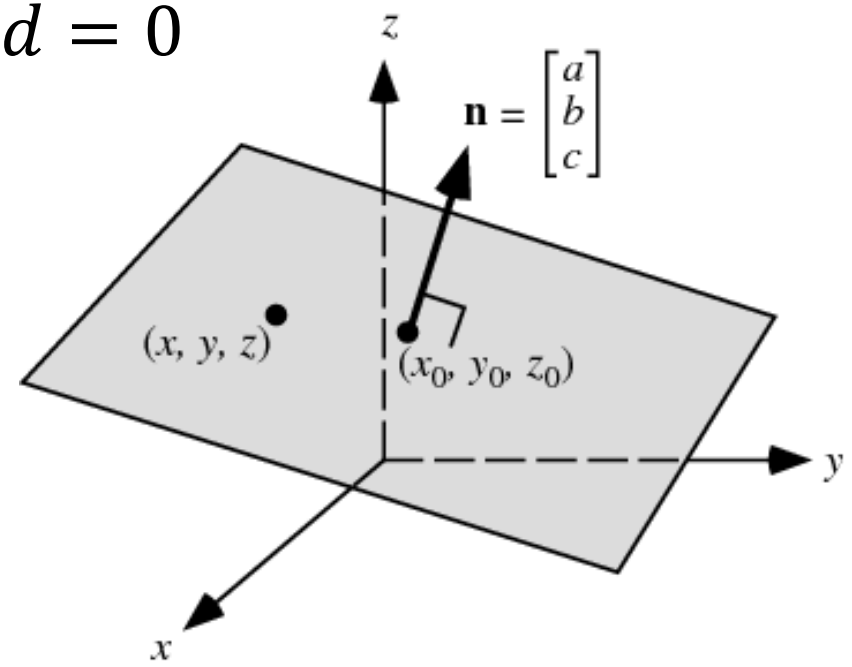


Equation of Planar Surfaces

$$a \cdot (x - x_0) + b \cdot (y - y_0) + c \cdot (z - z_0) = 0$$

Or:

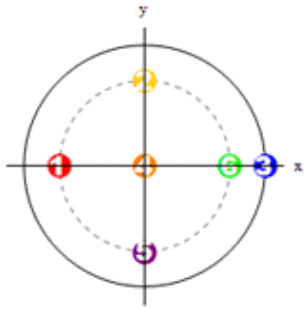
$$a \cdot x + b \cdot y + c \cdot z + d = 0$$

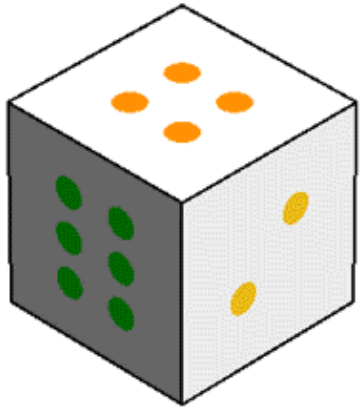


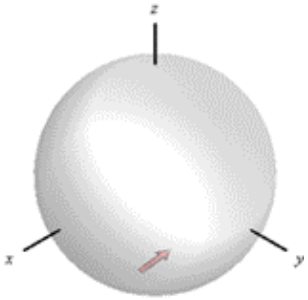
Transformation of Planar Surfaces in 3D Space

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = R_{3x3} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = T_{3x1} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$







Equation of Spherical Surfaces

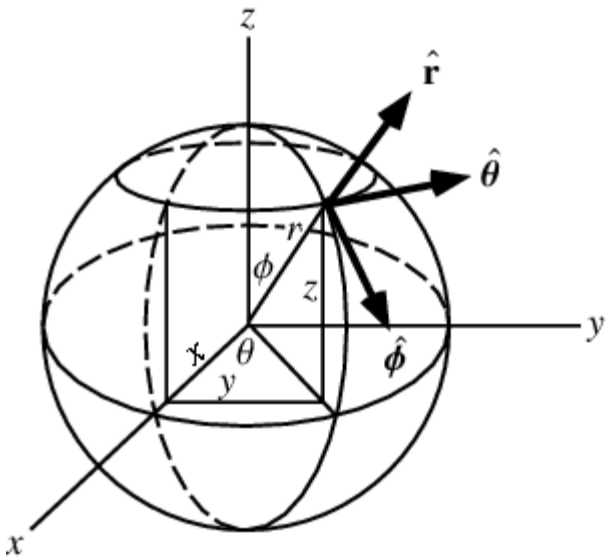
$$\frac{(x-x_0)^2}{R_x^2}+\frac{(y-y_0)^2}{R_y^2}+\frac{(z-z_0)^2}{R_z^2}=1$$

Or:

$$x = x_0 + R\cdot\cos(\theta)$$

$$y = y_0 + R\cdot\sin(\theta)$$

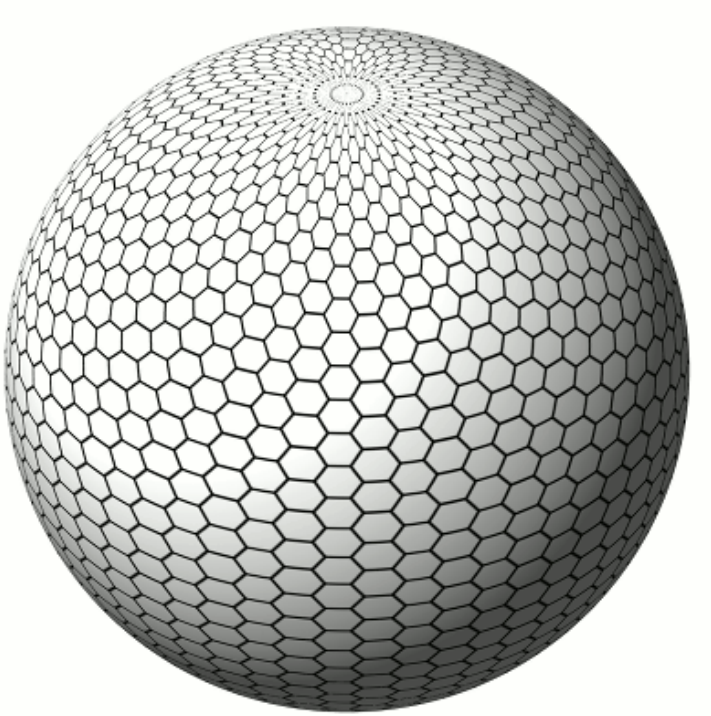
$$z = z_0 + R\cdot\cos(\varphi)$$



Transformation of Planar Surfaces in 3D Space

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = R_{3x3} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

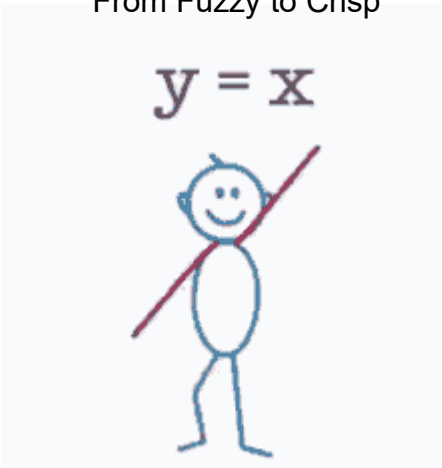
$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \end{bmatrix} = T_{3x1} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

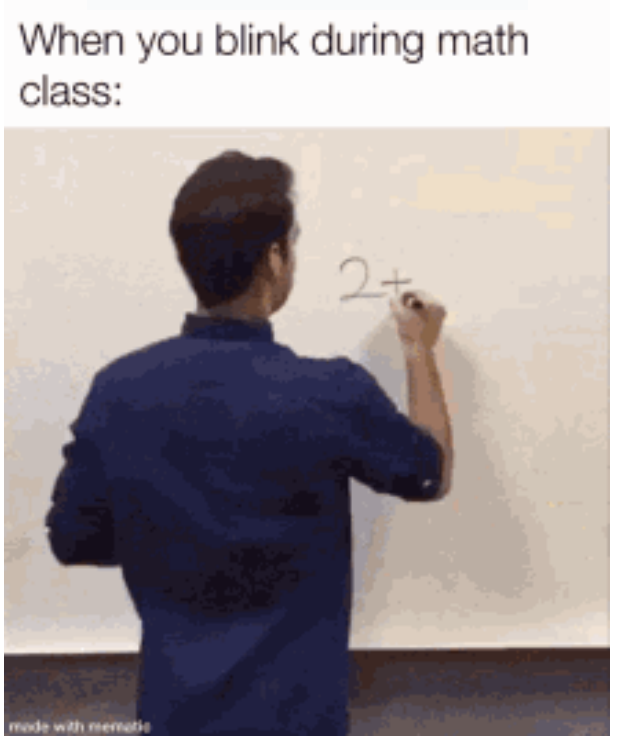


Summary of Lecture 2

- Invention of Technical Language
- Representation of Vectors
- Representation of Matrices
- Representation of Lines
- Representation of Curves
- Representation of Surfaces

From Fuzzy to Crisp





Outline of Module 2

- Lecture 1:
  - Use of Natural Language
- Lecture 2:
  - Use of Technical Language
- Lecture 3:
  - Use of Programming Language

NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical & Aerospace Engineering

Design, Machine, Control, Intelligence

Lecture 3 of Module 2

AI 3.0

MA4829 Machine Intelligence

Use of Programming Language to Represent Knowledge

XIE Ming, PhD (France)

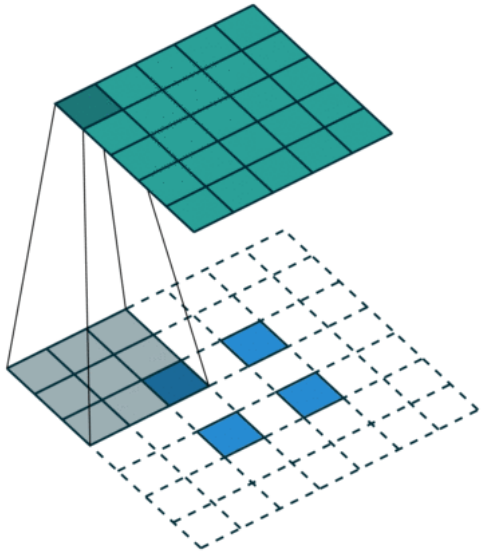
http://personal.ntu.edu.sg/mmxie

"We are living inside an ocean of signals"

What is a system with intelligence inside?

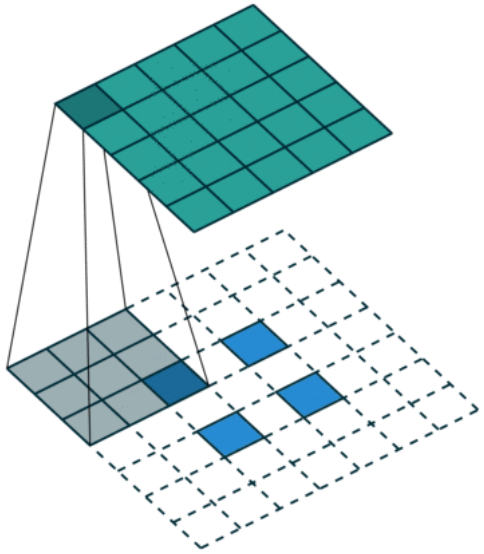
Outline of Lecture 3

- Invention of Programming Language
- C-Like Programming Languages
- Use of Programming Languages



Outline of Lecture 3

- Invention of Programming Language
- C-Like Programming Languages
- Use of Programming Languages



Human Languages Include Three Types:

- Natural Languages:
  - Spoken Languages (Natural Languages)
  - Written Languages (Natural Languages)

- Technical Languages
  - Mathematics

- Programming Languages
  - C
  - C++, C#, Java, …
  - Python, etc

They are the extensions to Natural Languages

What are programming languages?

Answer:

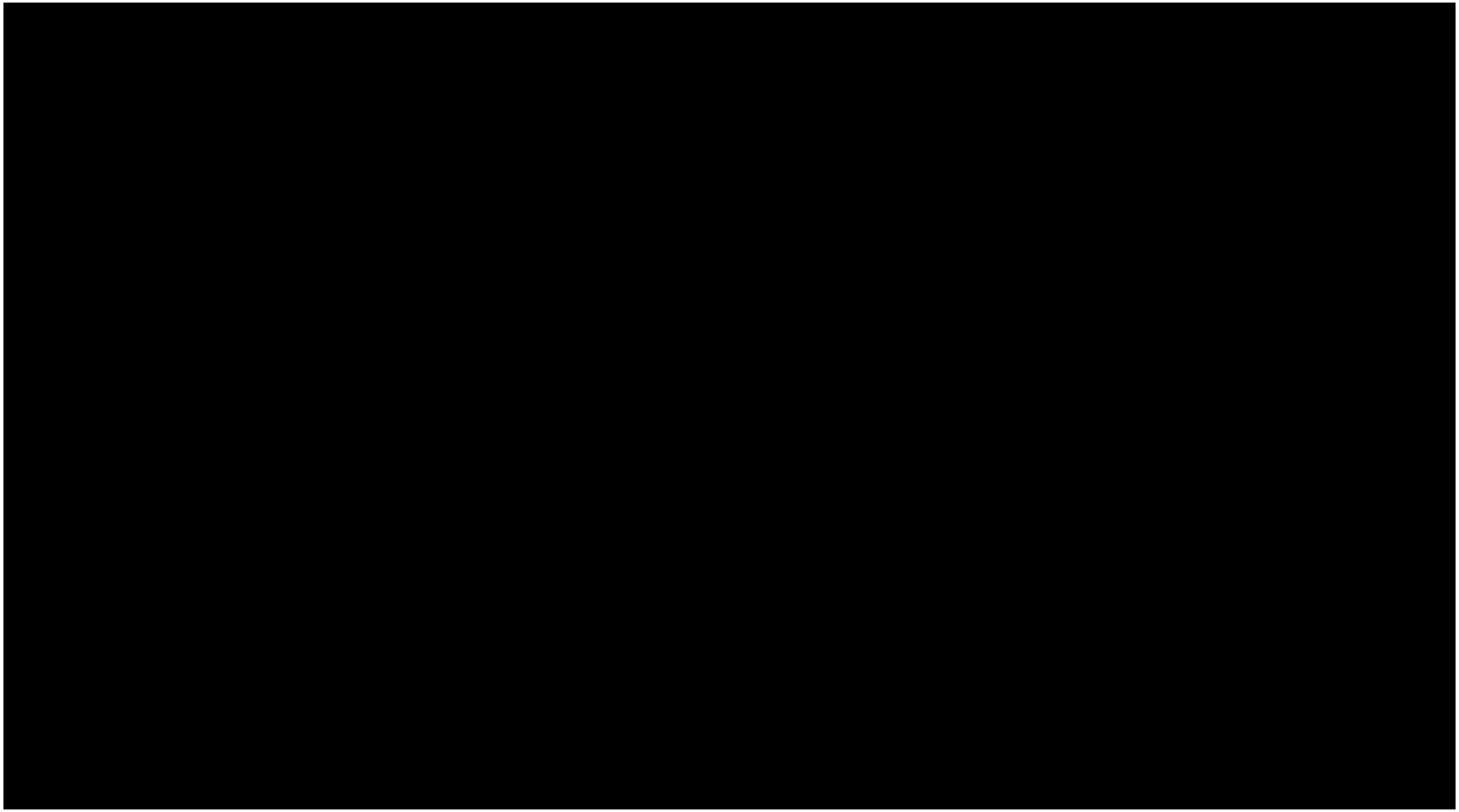
- They are the extension of natural languages
- They are in the forms of instructions and functions, which facilitate the translation of knowledge flows (e.g., algorithms) into executable programs by computers.

Why to invent programming languages?

- Knowledge could be perceived.
- Knowledge could be represented.
- Knowledge could be computed manually.
- Knowledge could be computed automatically.

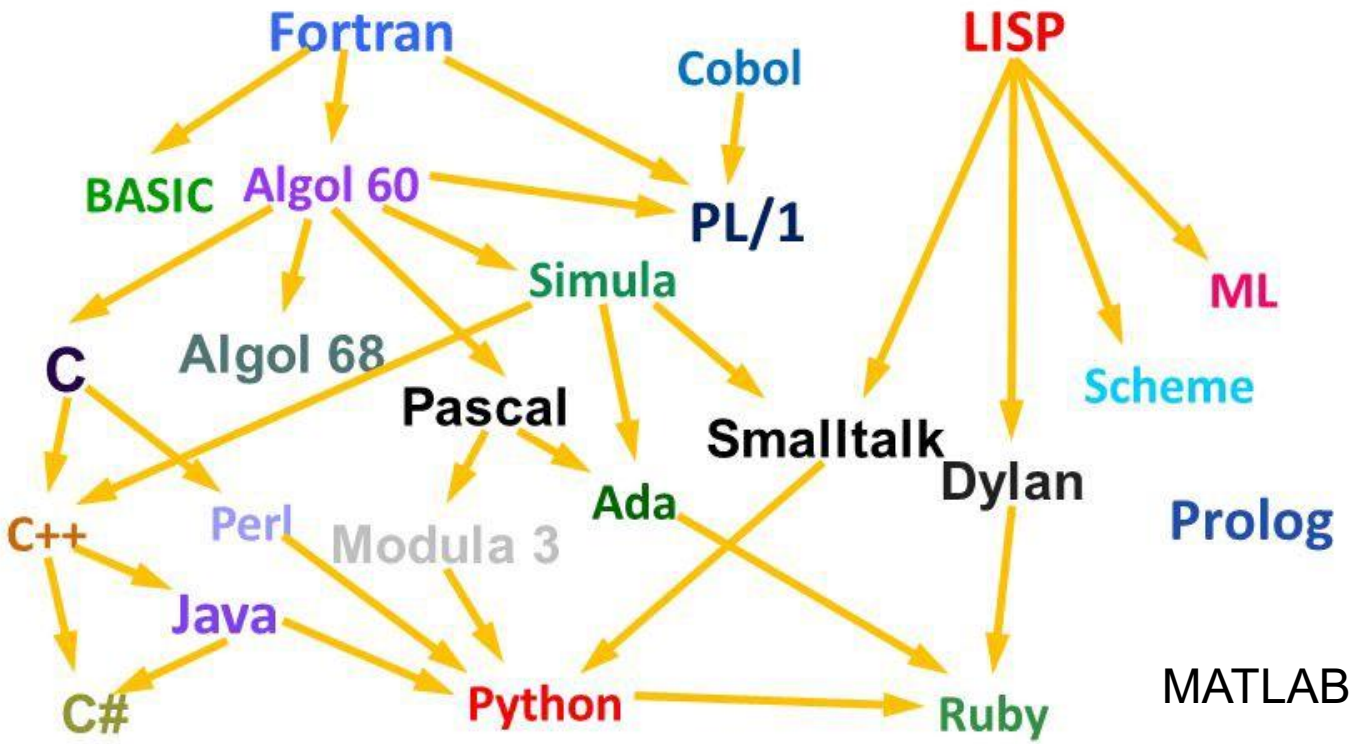
Early History of Programming Languages ...

Popular Programming Languages ...



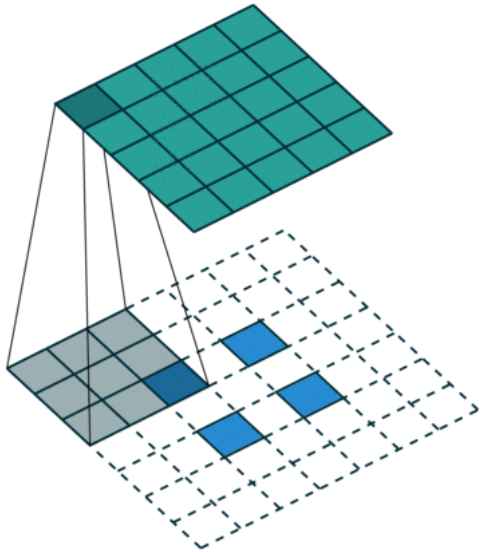
A Family Tree of Programming Languages

Some of the 2400 + programming languages



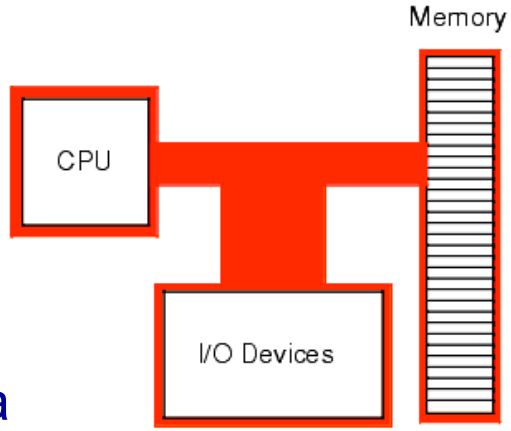
Outline of Lecture 3

- Invention of Programming Language
- C-Like Programming Languages
- Use of Programming Languages



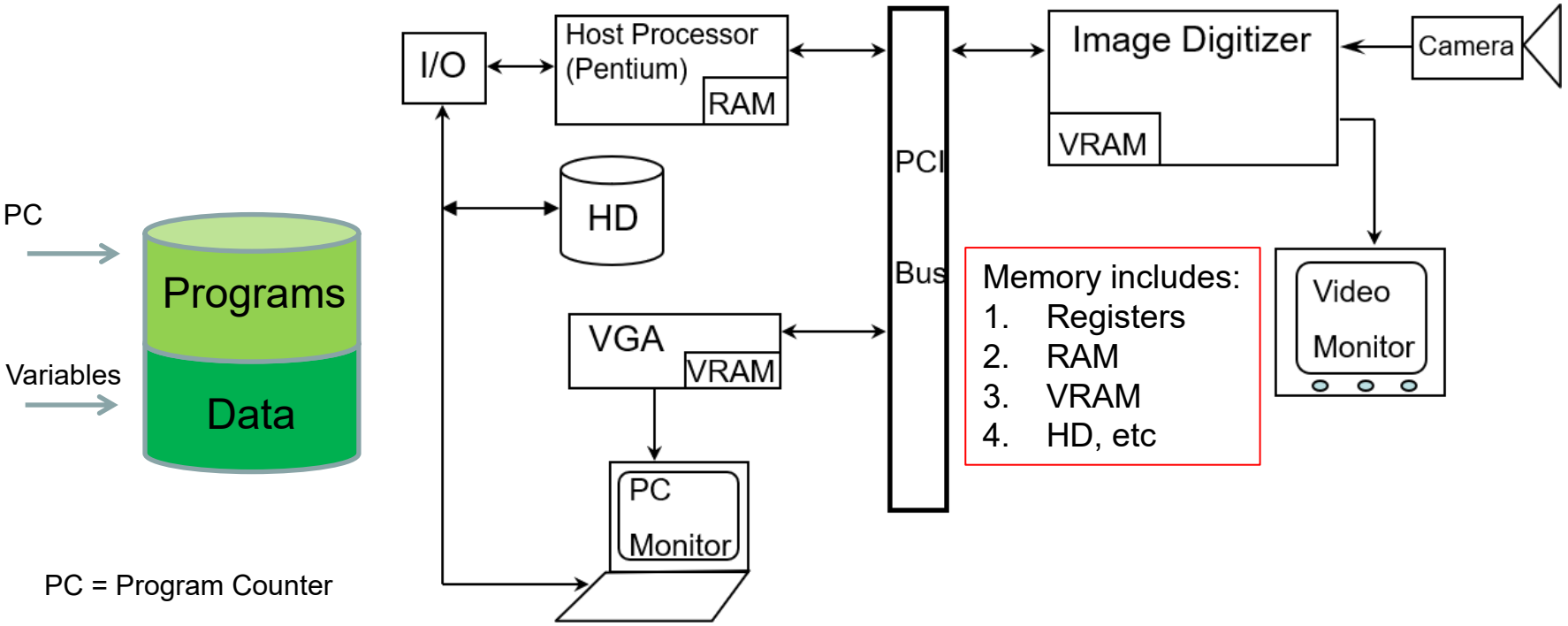
i love cp (computer programming)

Basics of Memory (1)



- Memory is a place in which we store raw data, processed data and programs.

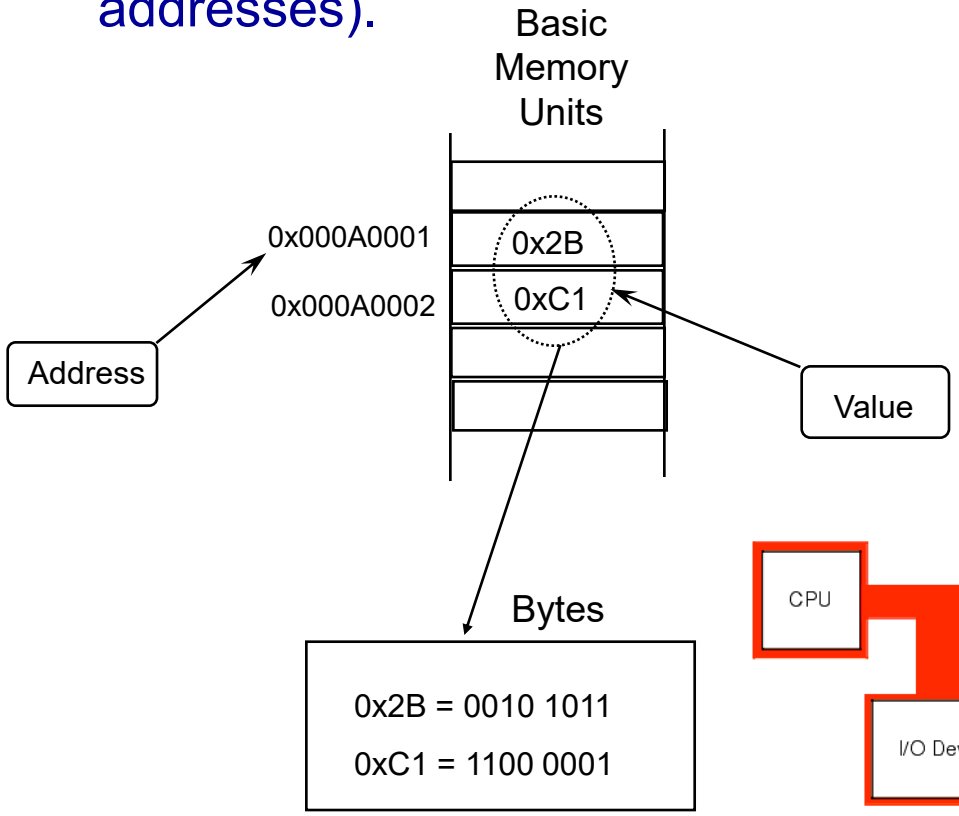
- Memory is generally divided into two types: a) data memory, b) program memory.

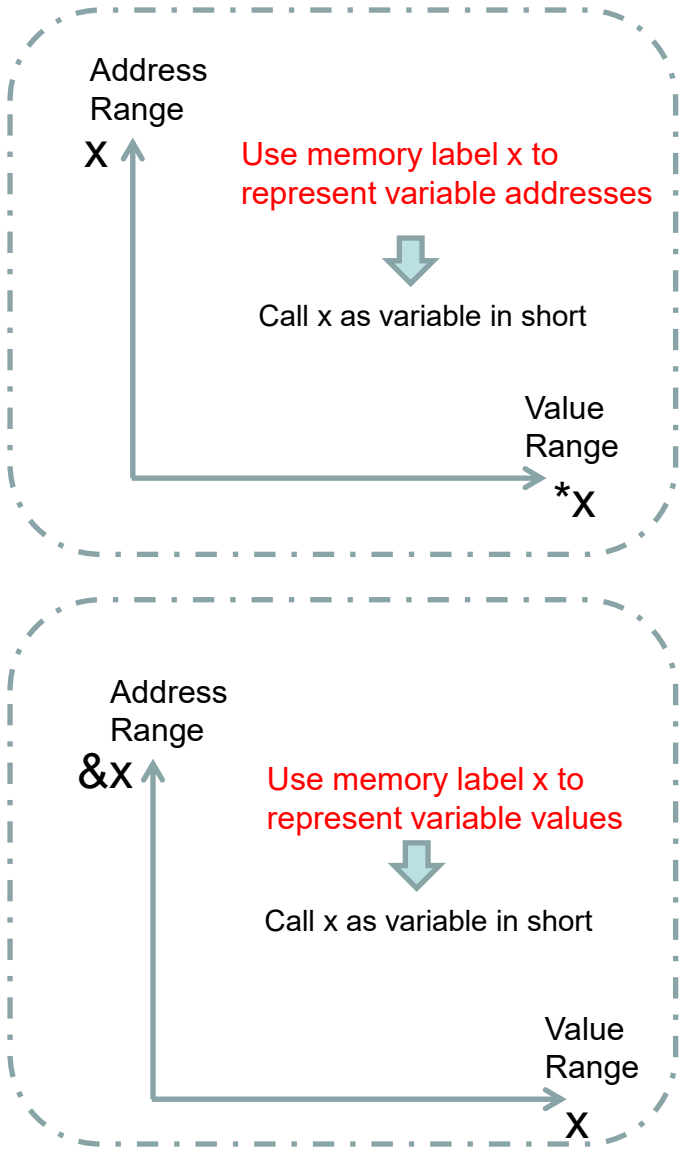


PC = Program Counter

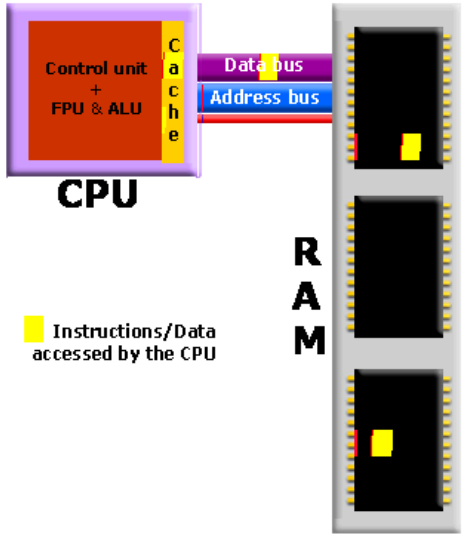
Basics of Memory (2)

- Physically, a memory is a one-dimensional vector of basic memory units (i.e. bytes).
- Conceptually, a memory is a two-dimensional space of knowledge (i.e., labels, values, addresses).

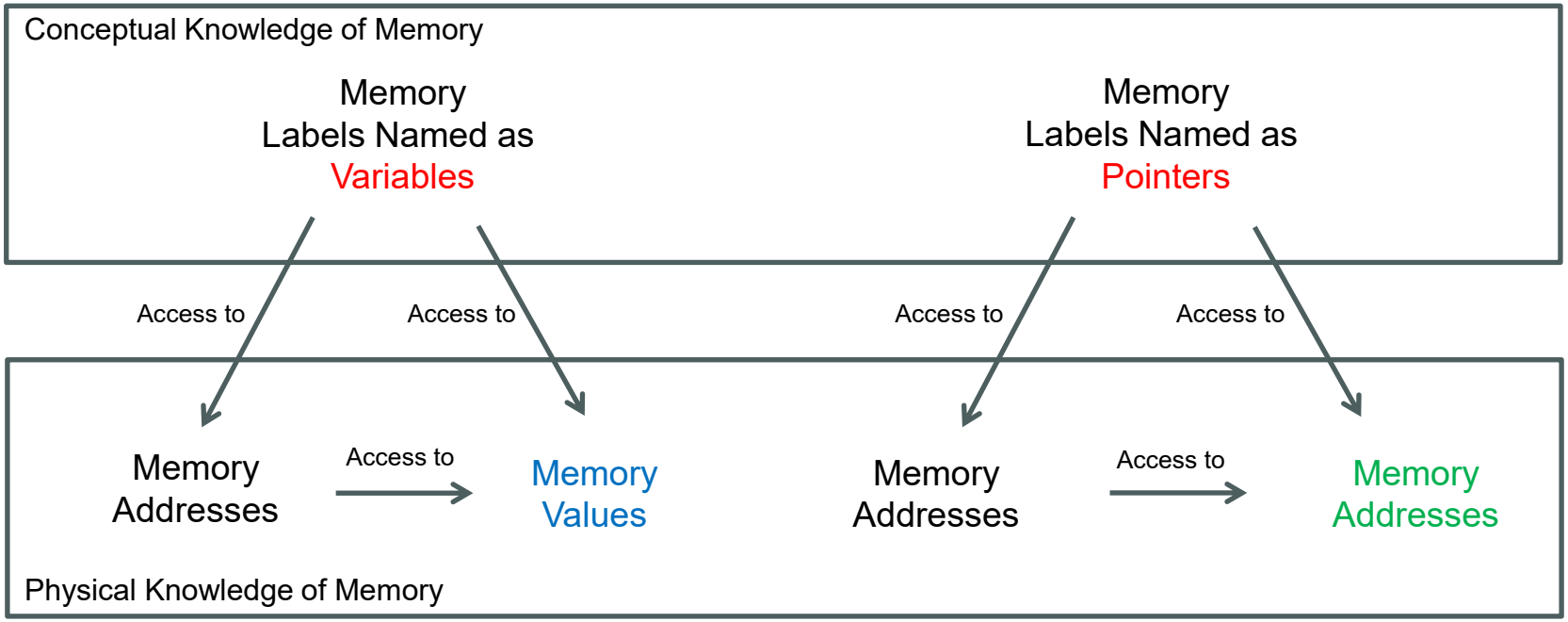




Basics of Memory (3)

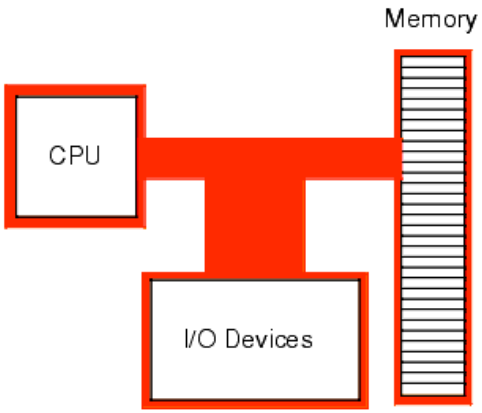


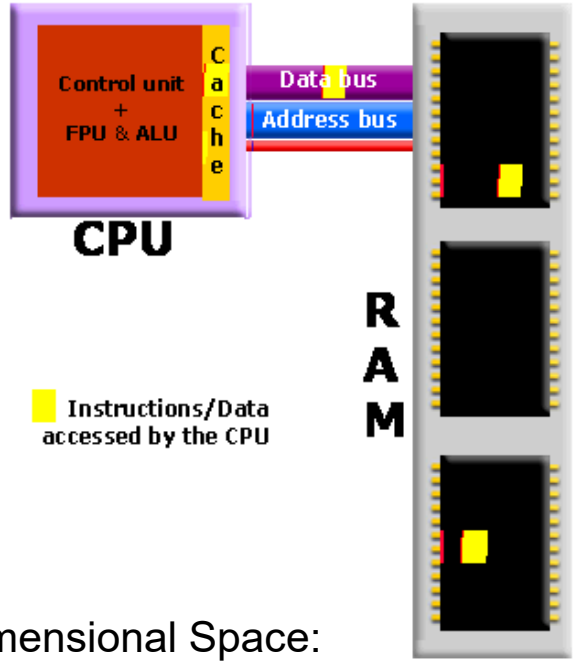
- Each memory unit has three knowledge: 1) conceptual knowledge named as “memory label”, 2) physical knowledge named as “memory address”, and 3) physical knowledge named as “memory content” or “memory data”.
- There are two types of memory label:



How to declare and create memory labels which hold constants in C?

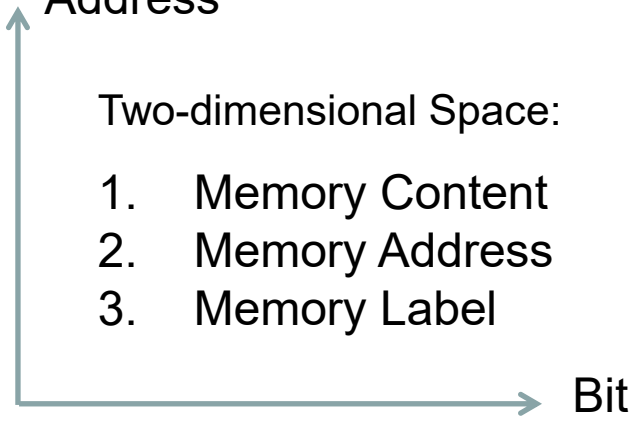
```
#define   WIDTH    512
#define   HEIGHT  512
```





Enumerated entries

```
enum Color  {Red, Green, Blue} ;

void  foo()
{
    Color   color_of_my_cloth;
    color_of_my_cloth = Red ;
}
```

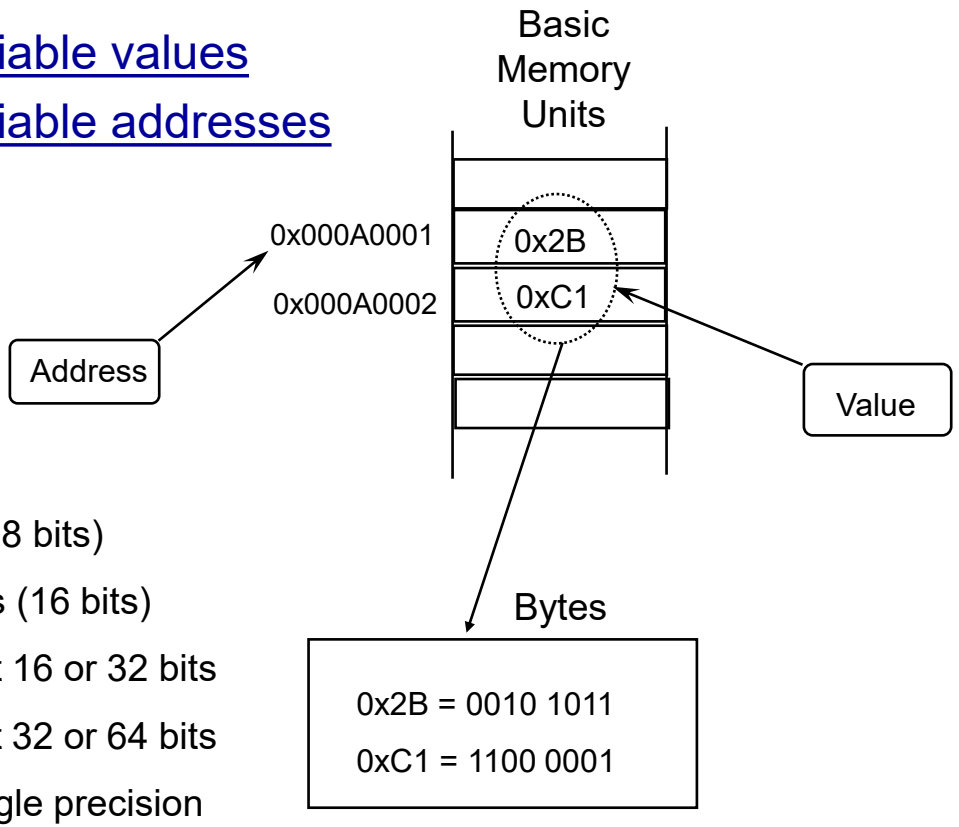


Two-dimensional Space:

1. Memory Content
2. Memory Address
3. Memory Label

How to declare and create memory labels to represent variable values/addresses in C?

- Declare memory labels to hold variable values
- Declare memory labels to hold variable addresses (pointers)




```
int       x, y, *a ;

double    z, *b ;
```

char	:	character (1 Byte or 8 bits)
short	:	an integer of 2 Bytes (16 bits)
int	:	an integer of at least 16 or 32 bits
long	:	an integer of at least 32 or 64 bits
float	:	a real number of single precision
double	:	a real number of double precision

(the qualifier “unsigned” can be applied for an integer)

How to assign numbers to memory labels?

a) directly

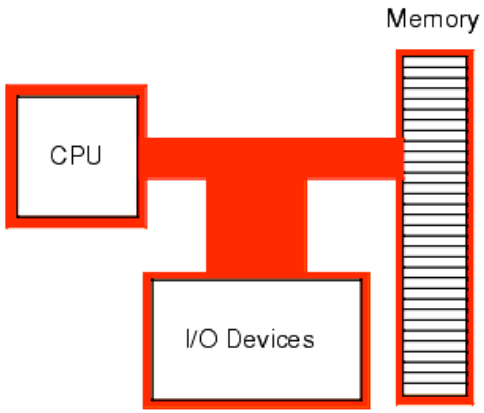
```
void foo()
{
    int    x, y ;

    x = 2 ;

    y = 10 ;
}
```

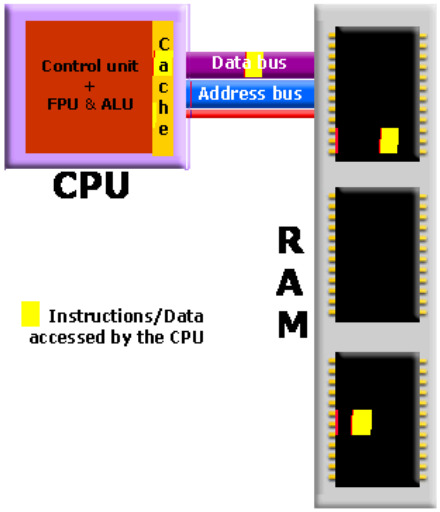
b) by arithmetic expression

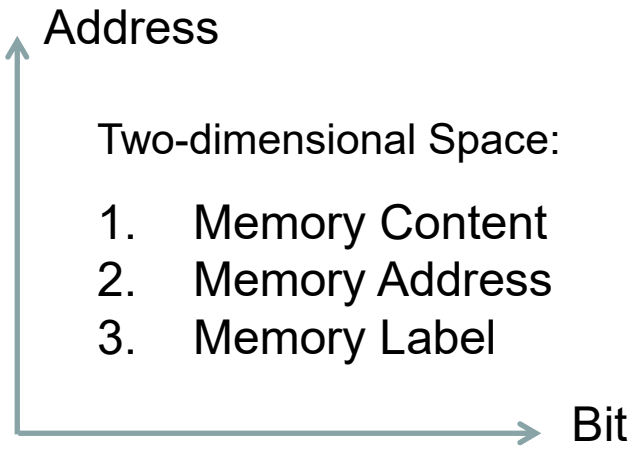
```
void foo()
{
    int    x, y, z ;

    x = 2 ;
    y = 10 ;
    z = (x*x + y*y) ;
}
```

c) by a function call

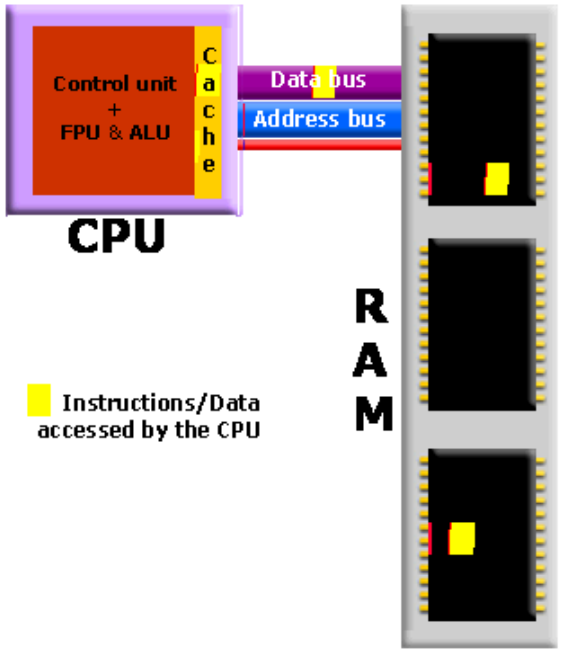
```
void foo()
{
    int    x, y, z ;

    x = 2 ;
    y = 10 ;
    z = sqrt(x*x + y*y) ;
}
```







How to declare and create functions in C programming language?

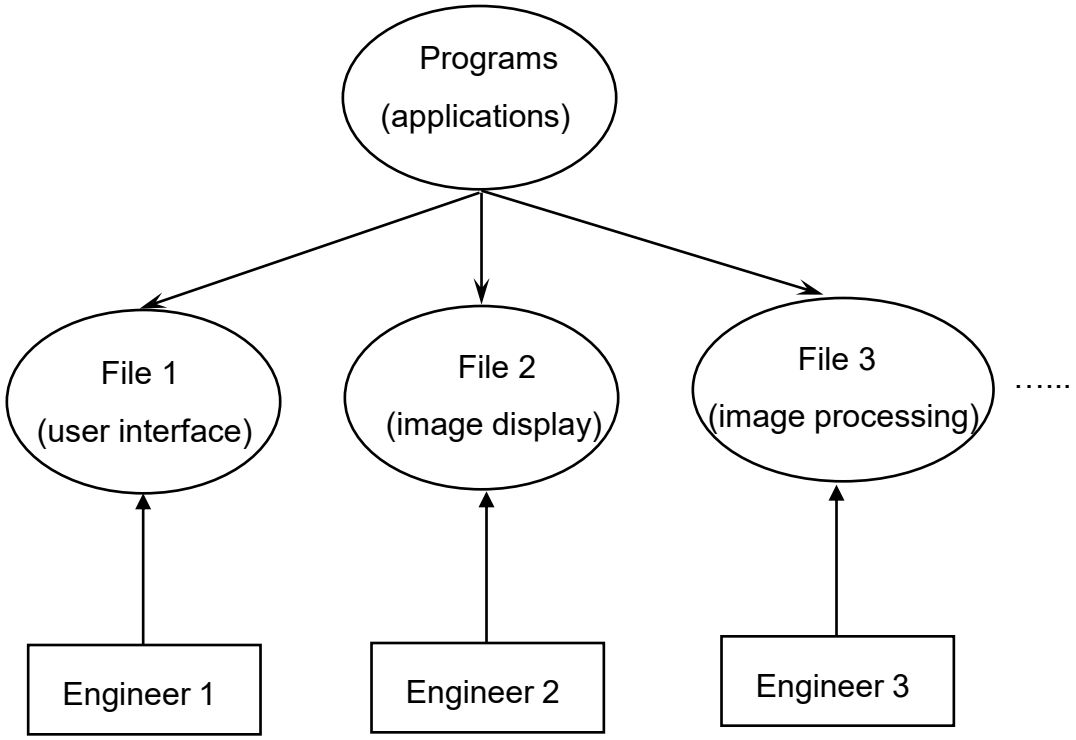


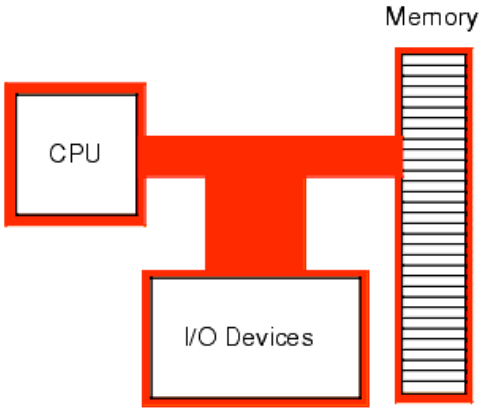
- A function has:
  - Name (i.e. also a memory label which holds an address)
  - Input
  - Output
  - Body of Instructions

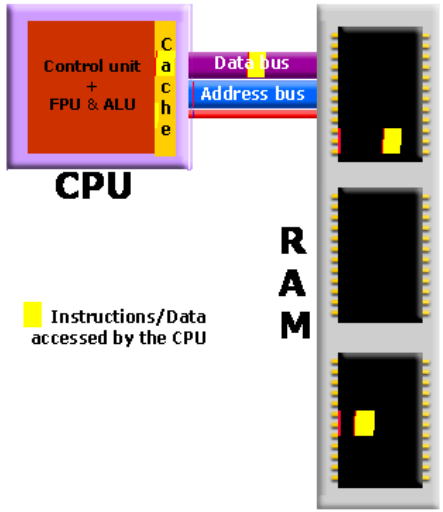
```
void PrintName(input)
{
    printf("\n> my name is XM") ;
}
```

```
int GetStudentNumber(input)
{
    int     nb_student ;
          ……
    return nb_student ;
}
```

If a program is very long, it is wise to functionally divide it into a set of files:







Problem:

How to handle the visibility of memory labels and functions across the files ?

How to handle the visibility of memory labels and functions across the files in C ?

- The qualifier "static" indicates that memory labels or functions are visible within a same file.
- The qualifier "extern" indicates that memory labels or functions are visible across all files.
- By default, without any qualifier, a memory label is visible globally or just within a function.

File 1

```
int          x,  y ;
static  int  z ;

void f1()
{
    double   a ;
    a = 1.3 ;   z = 8 ; x=3;
}
```

File 2

```
extern int   x,  y ;
static  int   z ;
double      n ;

static void f2()
{
    n = 1.5 ; y = 2 ; z = 9;
}
```

File 3

```
extern int          x,  y ;
extern double       n ;

void f3()
{
    f1() ;
    n = 1.2 ; x = 10 ; y = 1;
}
```

For the purpose of preventing data interference or conflict, it is important to enforce the use of visibility constraints:

Rule 1

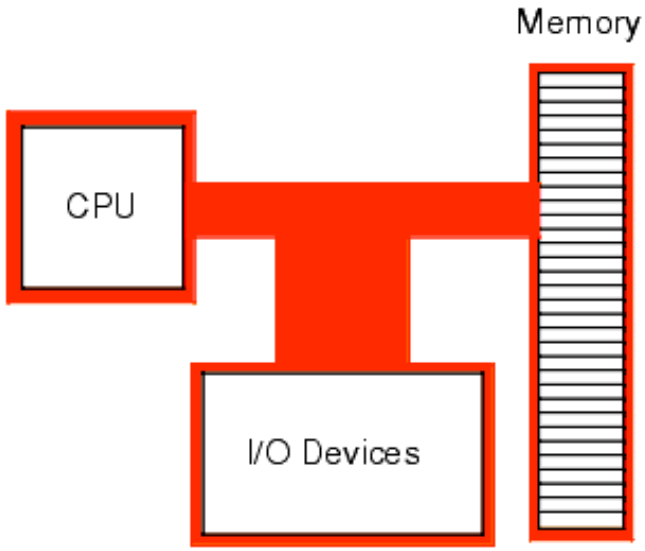
- By default, always declare the memory labels and functions with the qualifier “static” (this is to make them only visible within a file).

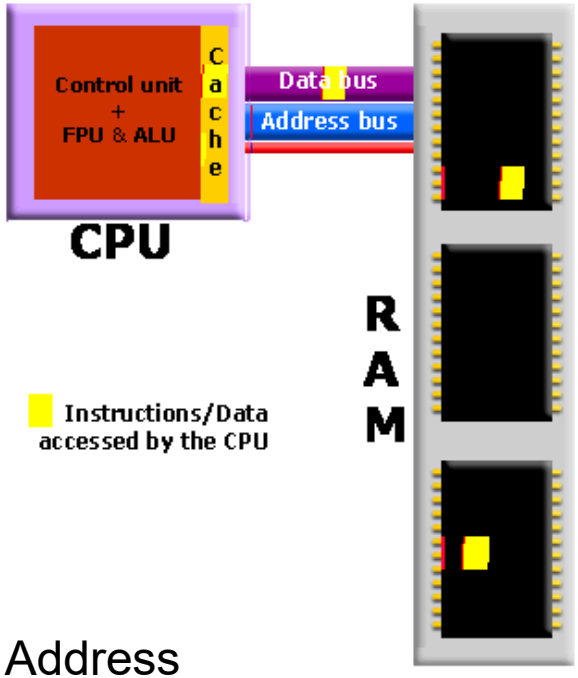
Rule 2

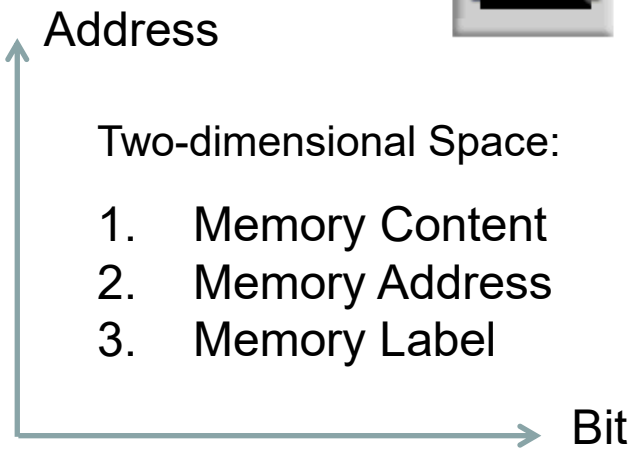
- If a memory label or function has to be visible across all files, use the qualifier “extern” explicitly in a conjugate header file such as “name.h”.

Example of Applying Rule 1

```
static int      x,   y ;

static   void   InitApplication()
{
    x = 0 ;   y = 0 ;
}

void   main()
{
        InitApplication() ;
}
```







Two-dimensional Space:

1. Memory Content
2. Memory Address
3. Memory Label

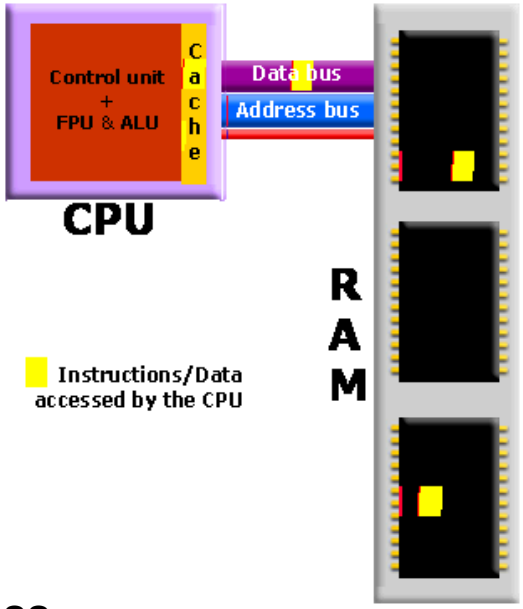
Example of Applying Rule 2

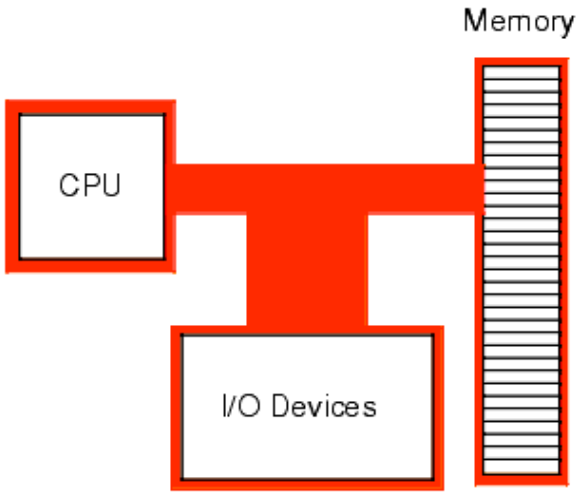
toto.c

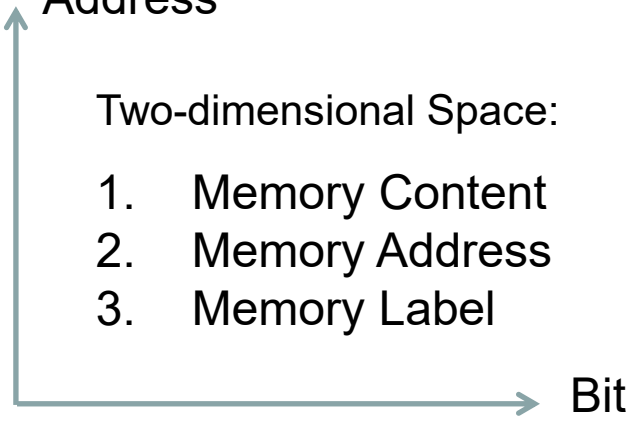
```
char lecture_name[100];

int GetStudentNumber()
{
    int   nb_student ;

    return nb_student ;
}

main()
{
    ….
}
```

toto.h

```
extern char lecture_name[100];

extern int GetStudentNumber() ;
```







More Examples …

whatiscoursename.c

```
#include “printcoursename.h”

int     subject_code ;

main(int argc, char **argv)
{
    subject_code = 4829 ;
    PrintCourseName() ;

    subject_code = 4825 ;
    PrintCourseName() ;

}
```

printcoursename.c

```
#include   <stdio.h>
#include   “whatiscoursename.h”


void  PrintCourseName()
{
    if (subject_code == 4829)
        printf(“\n Machine Intelligence”);
    else
    if (subject_code == 4825)
        printf(“\n Robotics”);
    else
        printf(“\n Unknown Course !”) ;
}
```

whatiscoursename.h

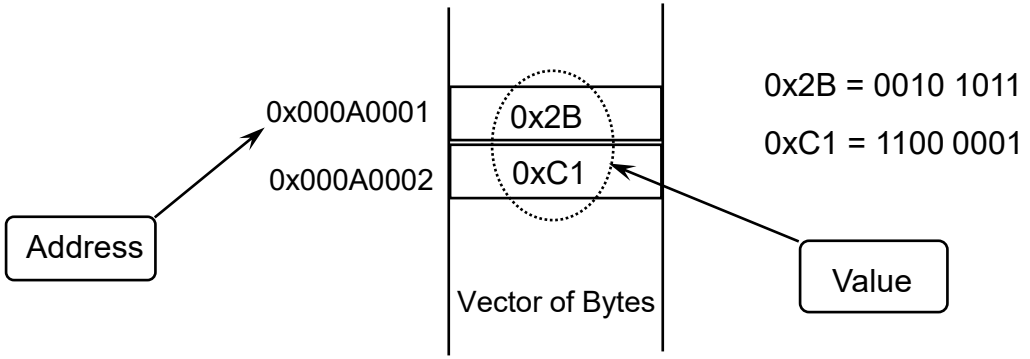
```
extern int     subject_code ;
```

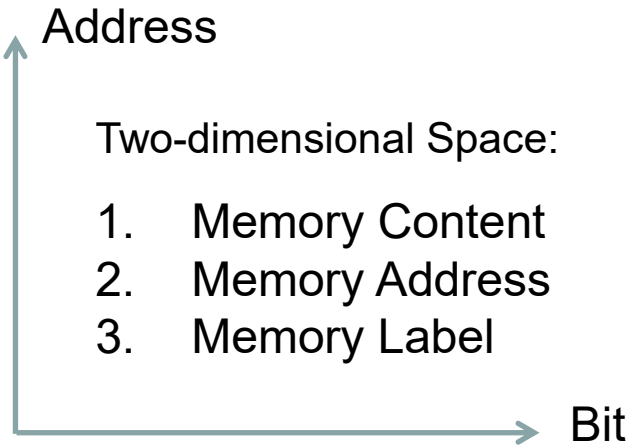
printcoursename.h

```
extern void     PrintCourseName() ;
```

Special Attention to: Pointer

- In C programming language, a pointer is a memory label which holds variable addresses. With a memory label, we could retrieve both the value and address, which are referenced by the memory label.






```
void  foo()
{
    int   x, *y, z  ;
    x = 0x2BC1 ;
    y = &x ;
    z = *y ;
}
```

In this example, the address of a memory label x is accessed by “&x” (i.e. ampersand x).

In this example, y is a memory label, which holds variable addresses. The value stored at address y is *y.

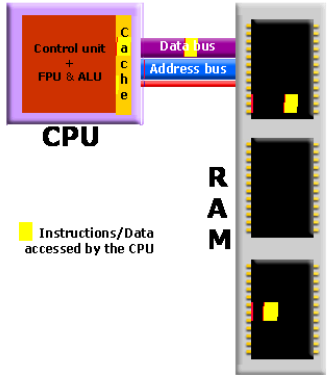
In this example, x=0x2BC1 while &x = 0x000A0001. So, we can do y = &x. When we do z = *y, then z = x.

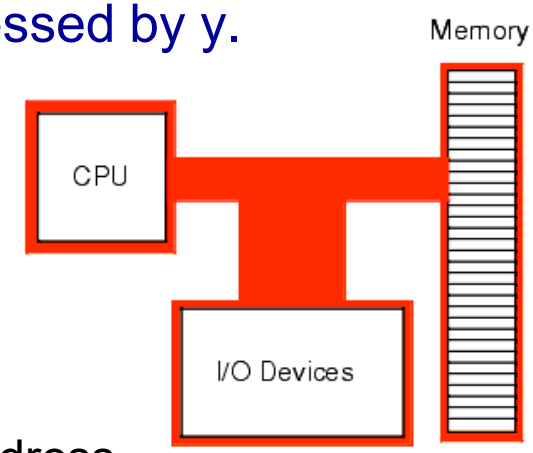
(NOTE: You may change “int” to “long” for today’s computers)

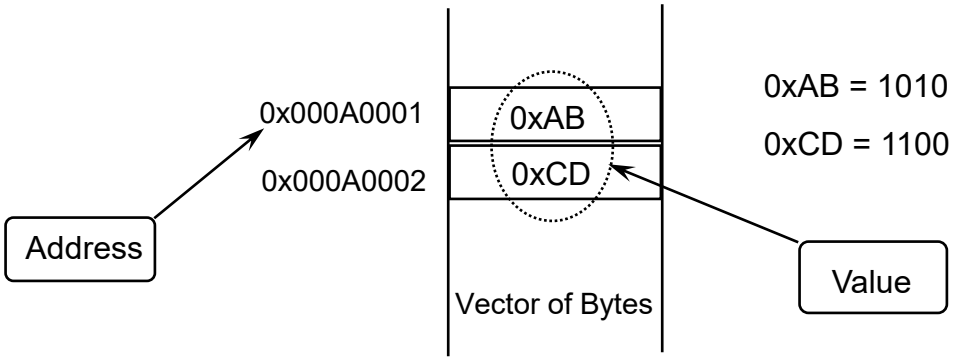
Example of Using Pointer in C

- y is a memory label which holds variable addresses.
- Address 0x000A0001 is assigned to y.
- Value 0xABCD is stored to the memory cell addressed by y.

```
void   foo()
{
   int    *y ;

   y = 0x000A0001 ;

   *y = 0xABCD ;
}
```

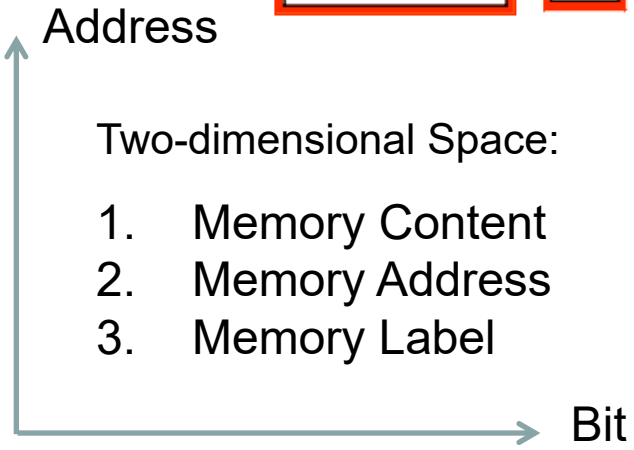






0xAB = 1010 1011

0xCD = 1100 1101



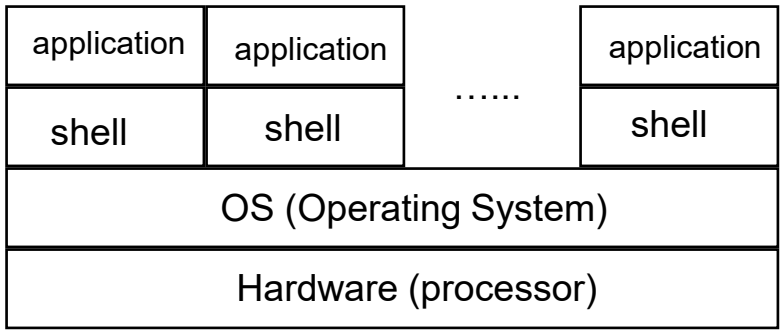
Two-dimensional Space:

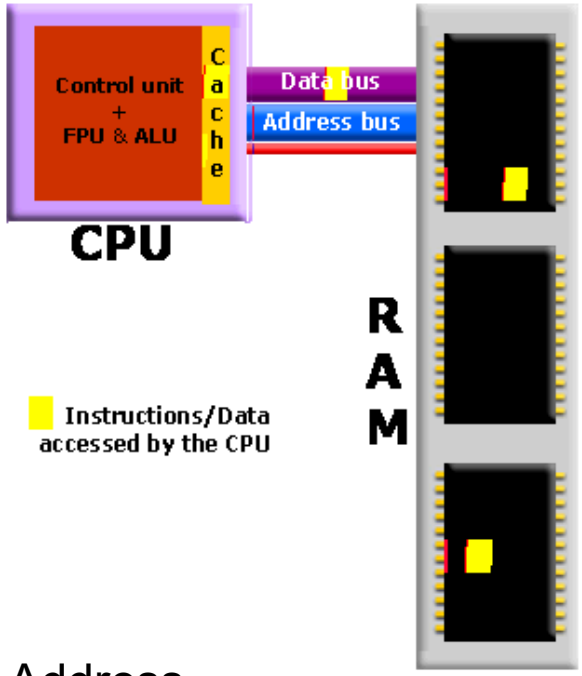
1. Memory Content
2. Memory Address
3. Memory Label

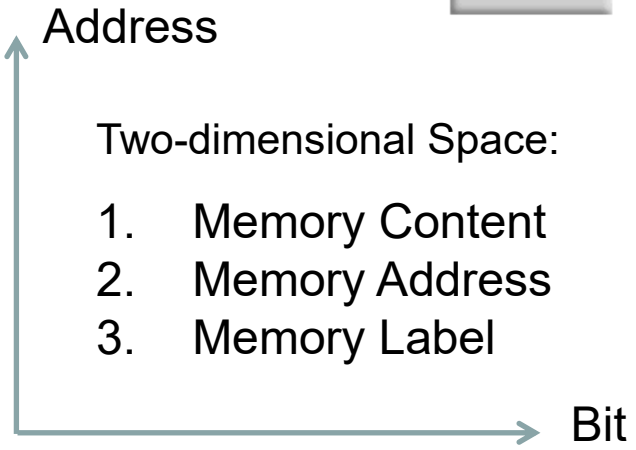
(NOTE: You may change “int” to “long” for today’s computers)

Entry Point of C Program

- A C-program always starts with the "main()" function !!!
- A function may have many arguments as input.
- The "main()" function is always activated by a "shell" program of the operating system:





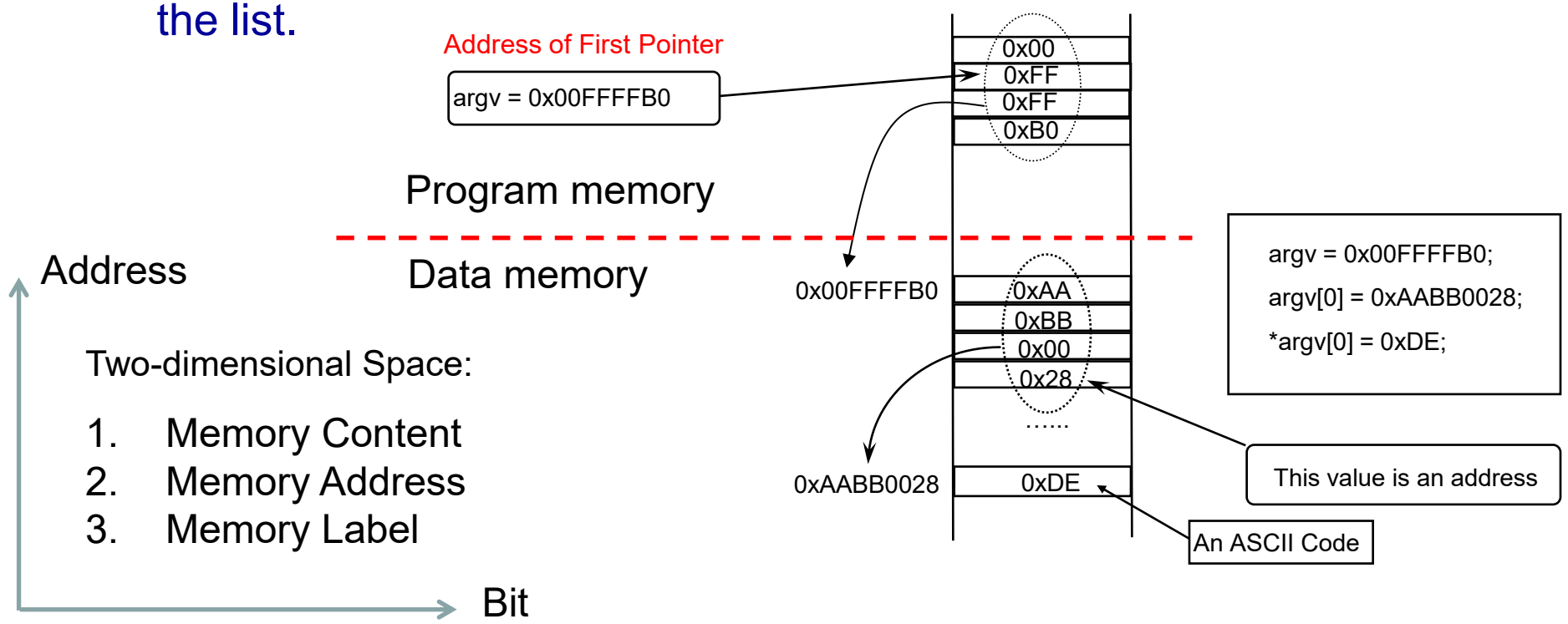


Entry Point of C Program (continued)

List of pointers

**argv = *argv[ ];

- What does the pointer inside "main(int argc, char **argv)" mean ?
- Memory label "argv" holds the first address of a list of pointers, each of which stores an address. "argv+1" holds the second address in the list.



How to translate the ASCII coded file(s) of a program into an executable binary file in C ?

a) To use Compiler to convert ASCII coded files into computer-readable files (i.e., binary or "object" files named in computer science).

(*.c --> *.o)

- Program Compilation
- Word Embedding/Encoding

b) To use Linker to combine the "object' files and the needed "object" files of invoked library functions into executable file.

(*.o + *.a --> executable file)

An archived file is a binary file in which reusable functions are pre-stored

Example

toto.c

```
#include <stdio.h>
#include <math.h>

main(int argc, char **argv)
{
    printf("\n The square root of 9 is: %f", sqrt(9.0)) ;
}
```

Step 1: Conversion of ASCII coded program file into "object" file:

INPUT: toto.c

OUTPUT: toto.o

COMMAND: cc -c toto.c

(this will generate toto.o)

Step 2: Linking of our object file(s) and the object file(s) of invoked function(s) into an executable file:

INPUT: toto.o and libm.a

OUTPUT: toto

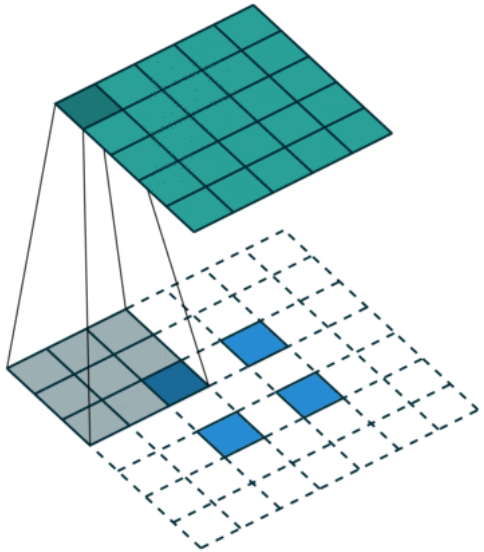
COMMAND: cc -o toto toto.o -lm

("-lm" means to link with object files inside library libm.a)

(the result will be an executable file named "toto")

Outline of Lecture 3

- Invention of Programming Language
- C-Like Programming Languages
- Use of Programming Languages



Programming Language versus Natural Language

Programming Language

- Memory labels (Variable values, Variables addresses, and Constant values, etc)
- Types of Labels
- Syntax
- Instructions/Expressions
- Functions
- Comments

Natural Language

- Words (Properties, Constraints, and Identities, etc)
- Types of Words
- Grammatical Rules
- Sentences
- Chapters
- Footnotes

Translation

Programming Languages Include Two Types:

- Instructing Languages: Languages for users to write programs with the help of instructions:
  - Reusable instructions
  - Sharable instructions
  - Libraries of instructions

```c
#include <stdio.h>
int main() {

    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    // calculate the sum
    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}
```

- Scripting Languages: Languages for users to write scripts with the help of commands:
  - Reusable commands
  - Sharable commands
  - Folders/Libraries of commands

```python
from PIL import Image
import os

input_folder = '/path/to/images'
output_folder = '/path/to/resized_images'
desired_size = (100, 100)

for filename in os.listdir(input_folder):
    with Image.open(os.path.join(input_folder, filename)) as img:
        img.thumbnail(desired_size)
        img.save(os.path.join(output_folder, filename))
```

Example of Program:

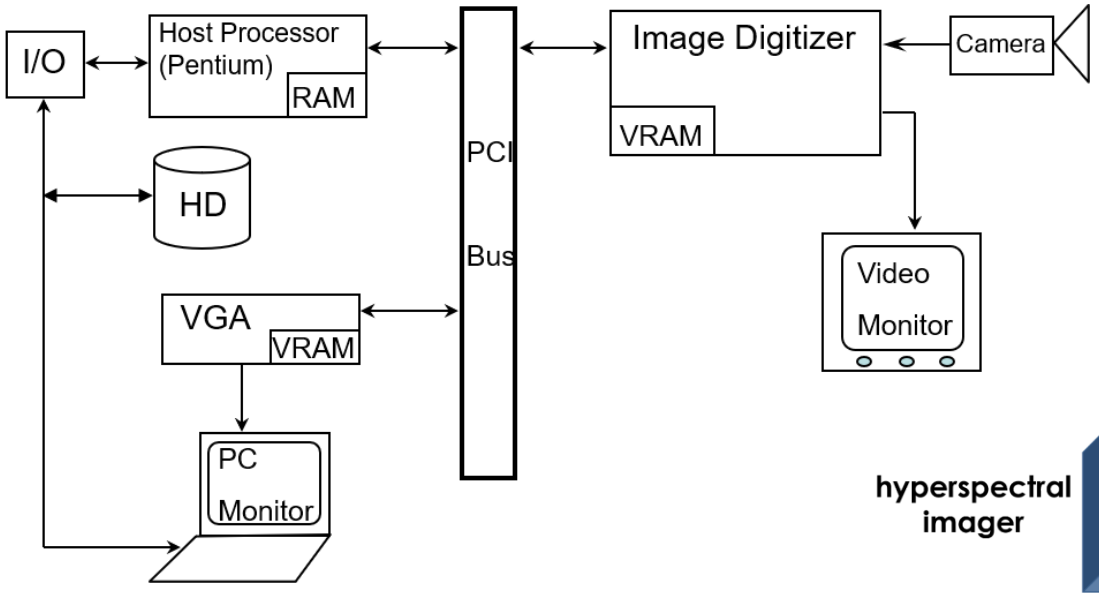
```c
#include <stdio.h>
int main() {

    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    // calculate the sum
    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}
```

Example of Script

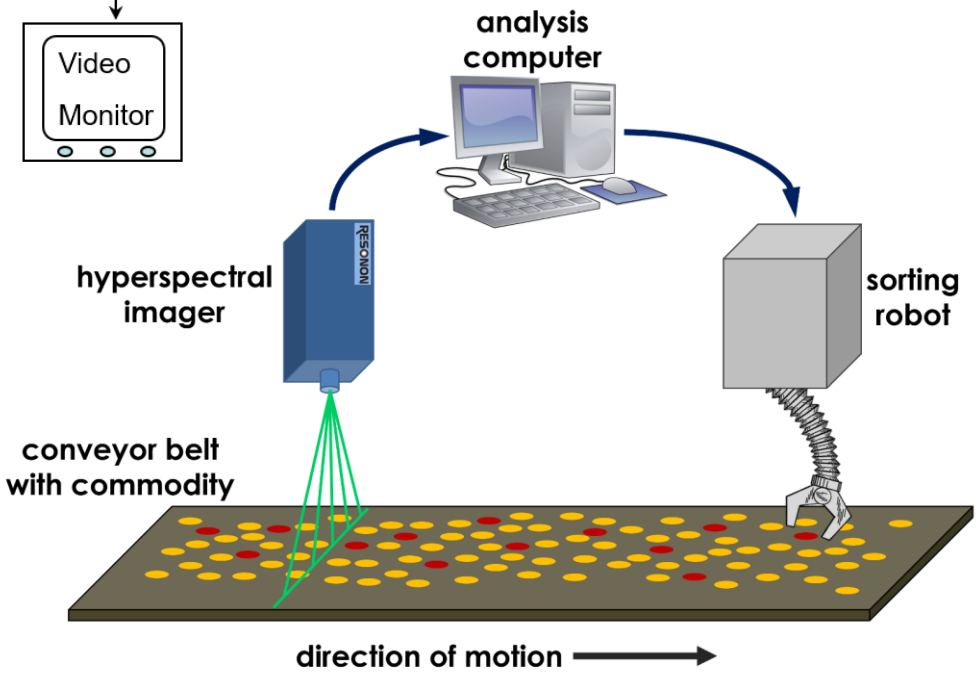
```
from PIL import Image
import os

input_folder = '/path/to/images'
output_folder = '/path/to/resized_images'
desired_size = (100, 100)

for filename in os.listdir(input_folder):
    with Image.open(os.path.join(input_folder, filename)) as img:
        img.thumbnail(desired_size)
        img.save(os.path.join(output_folder, filename))
```

1st Purpose of Using Programming Languages

- To computerize human intelligence in various domains



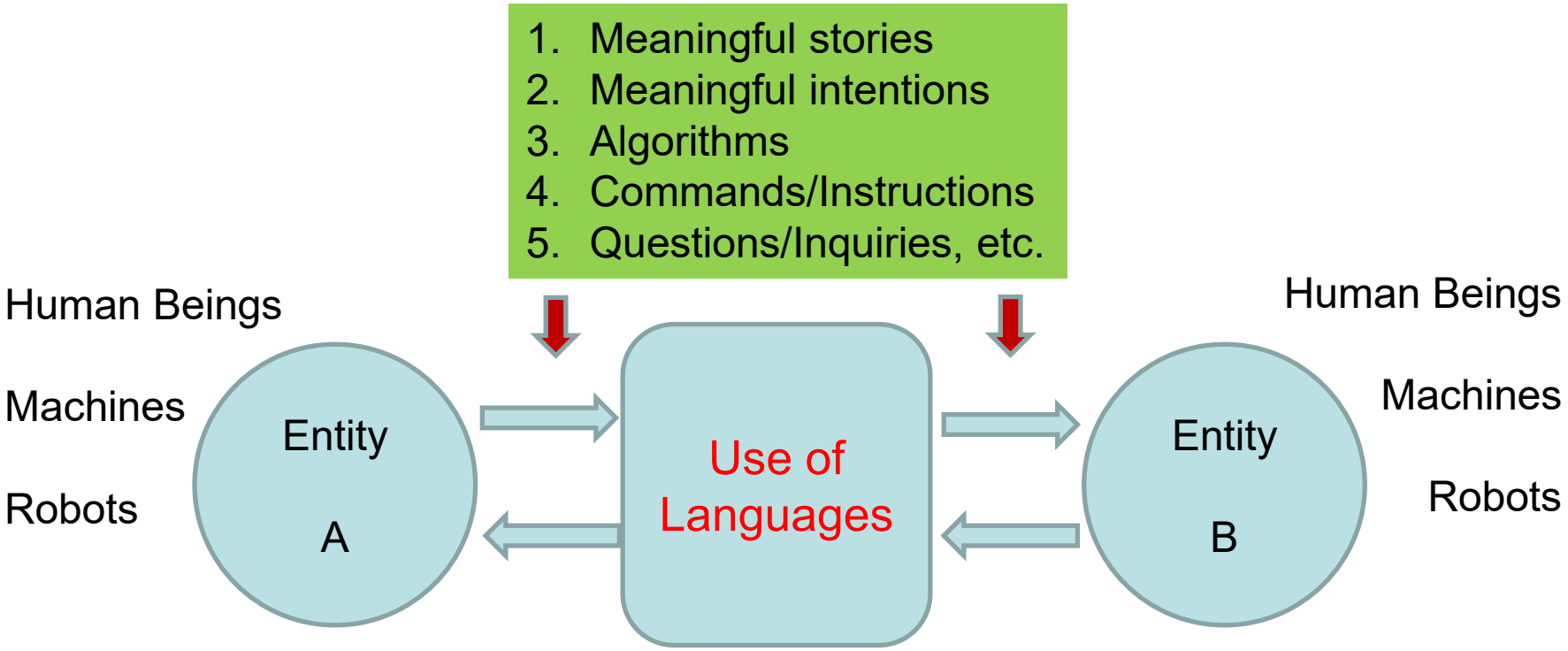
- Automatic Computations
- Automatic Data Processing
- Automatic Decision Making
- Automatic Control, etc.



Example: Motion Signals Enable Automation ...

2nd Purpose of Using Programming Languages

- To design and deploy machine self-intelligence for interaction/collaboration



"We are living inside an ocean of signals"

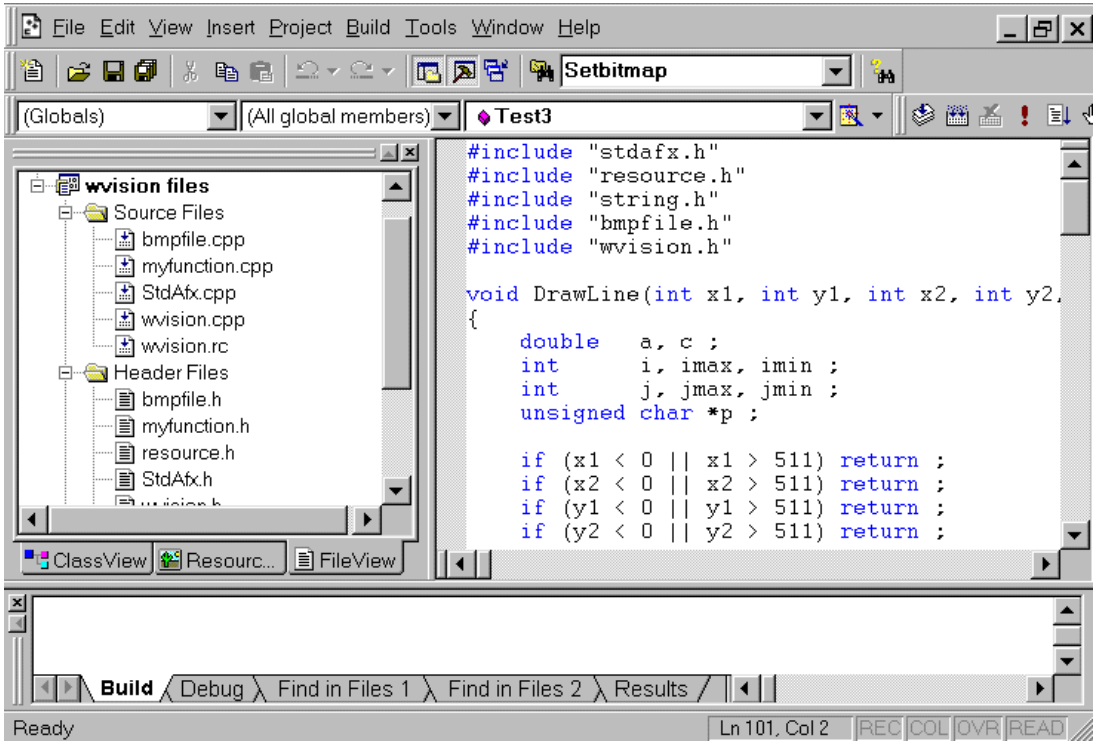
Example: Visual Signals Enable Autonomy ...

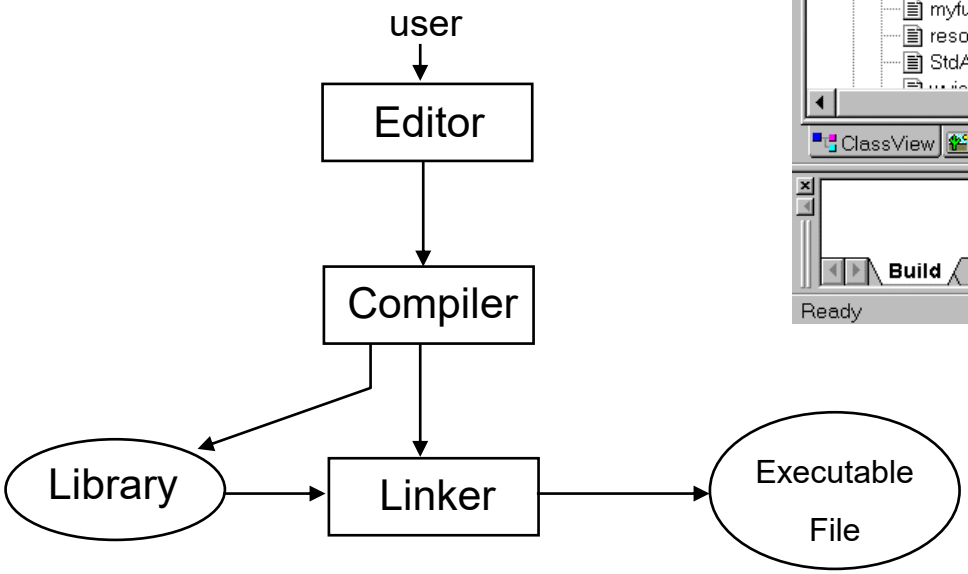
How to use a programming language?

ANSWER: To use IDE which includes:

- Editor
- Compiler
- Linker
- Loader
- Debugger

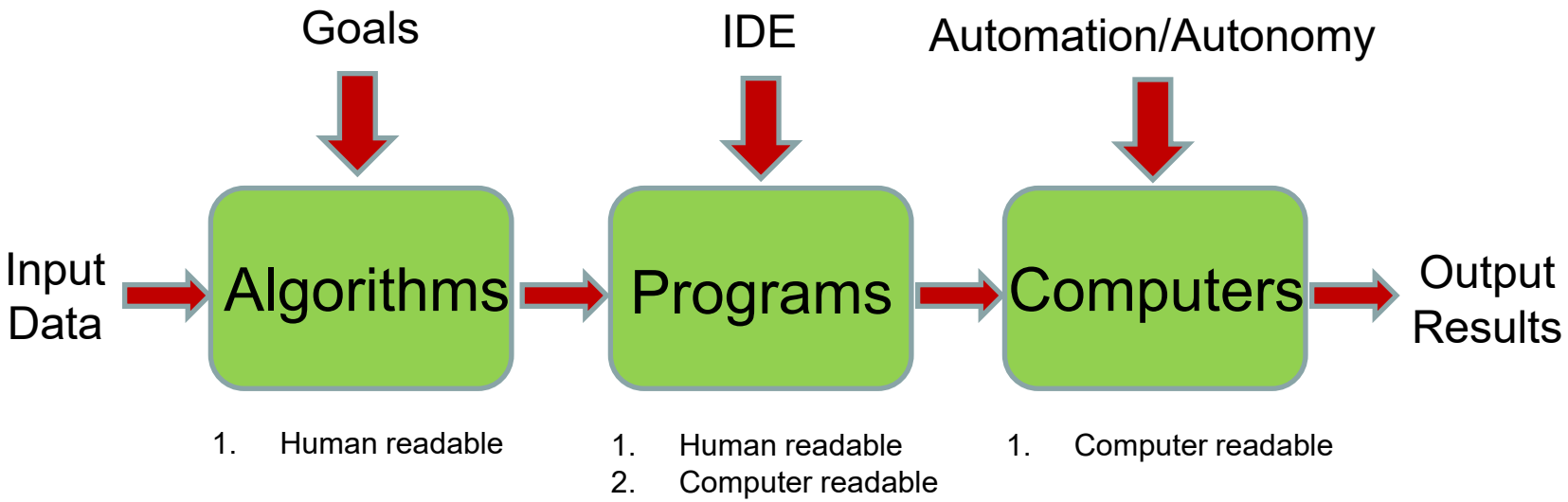
IDE: Integrated Development Environment





MATLAB has its own IDE

Procedure of Using Programming Languages in General



IDE: Integrated Development Environment

How to allocate memories? How to plan computations?

Procedure of Planning Computations

- Step 1: Describe your solution in the form of mathematics or logics.
- Step 2: Transform your solution into algorithms which consist of series of arithmetic and/or logical computations.
- Step 3: Draw flowcharts which interconnect arithmetic and/or logical computations in terms of their inputs and outputs.
- Step 4: Describe flowcharts with the use of a programming language.

Example of Transforming Solution into Algorithm

Solution:

$$ax^2 + bx + c = 0$$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Algorithm:

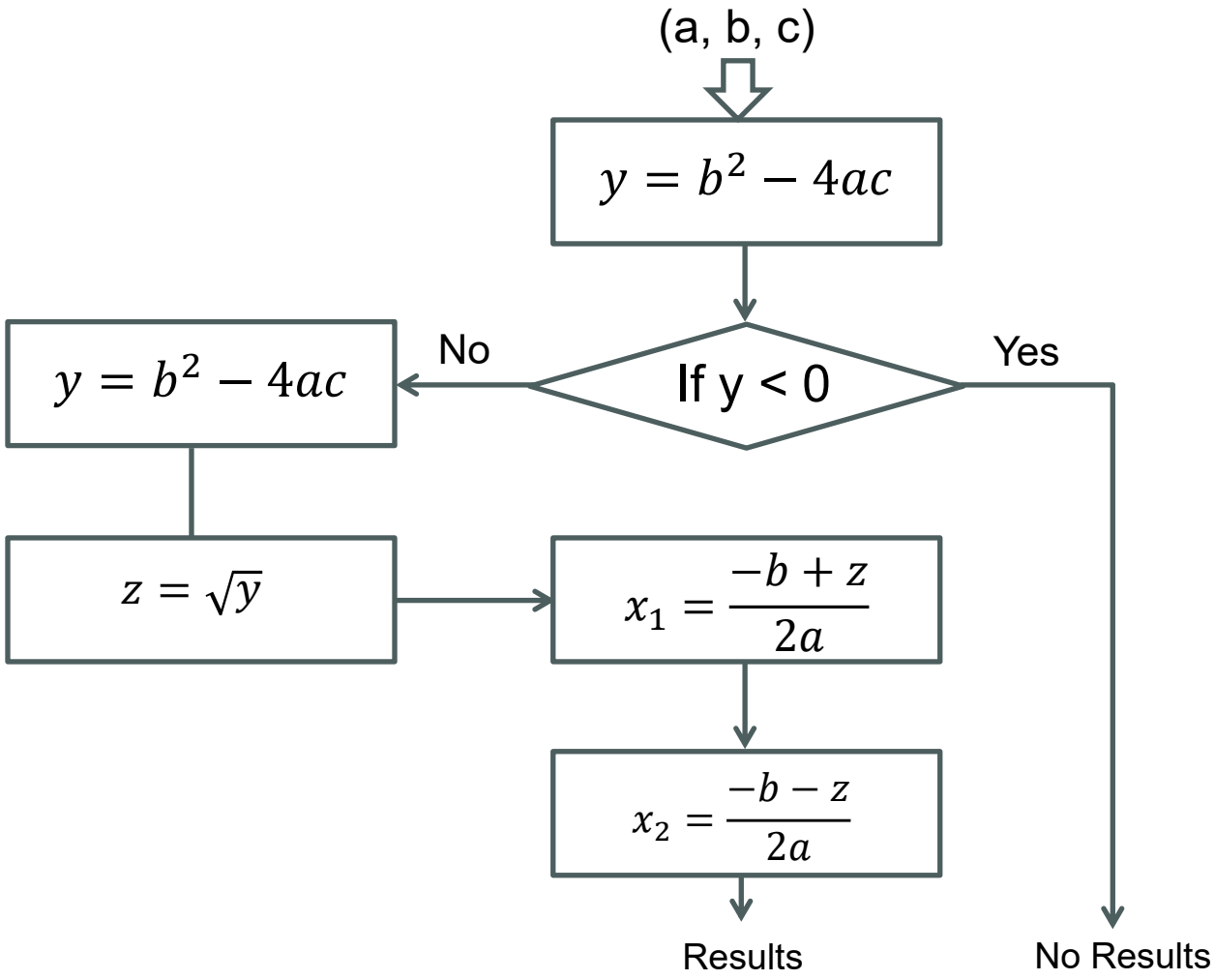
First do: $y = b^2 - 4ac$

Then do: $z = \sqrt{y}$

Then do: $x_1 = \frac{-b + z}{2a}$

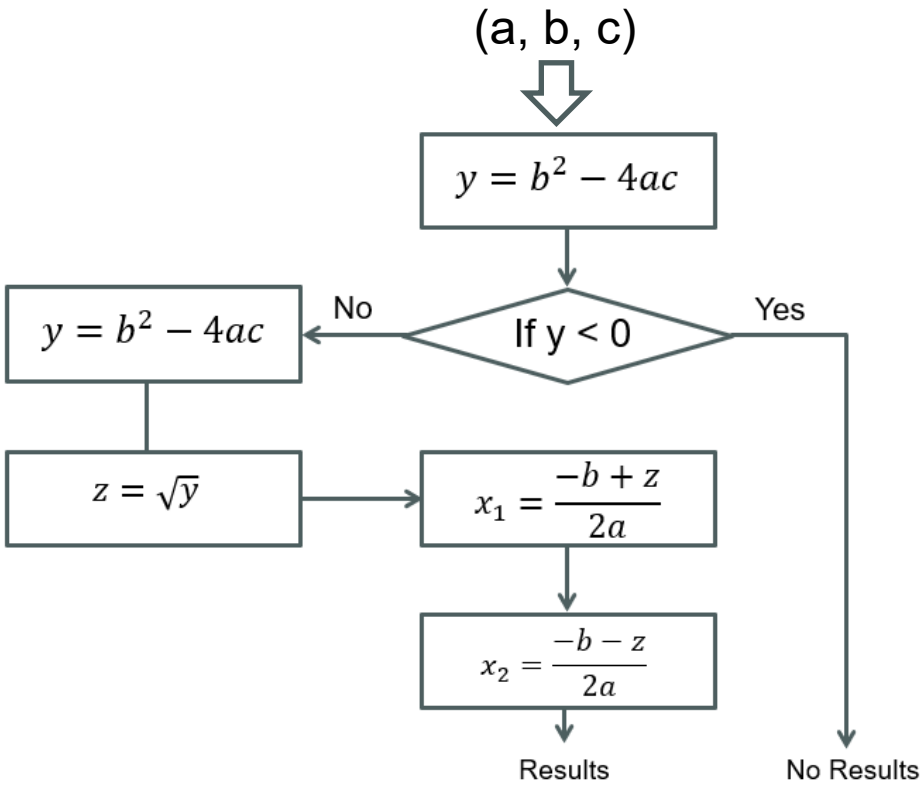
Then do: $x_2 = \frac{-b - z}{2a}$

Example of Translating Algorithm into Flowchart (i.e., Data Flow Diagram)



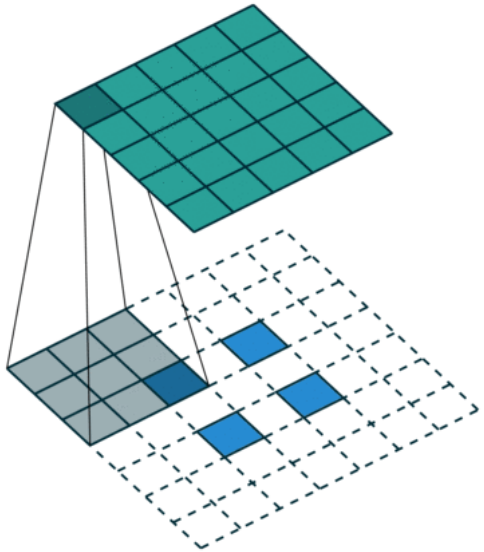
Example of Translating Flowchart into Program (i.e., Flow of Instructions)

```
void ComputeRoot(int a, int b, int c, double *x1, double *x2)
{
        int             x, y, z ;

        y = b*b – 4*a*c ;

        if (y >= 0)
        {
                z = sqrt(y) ;

                *x1 = (-b + z)/(2*a) ;

                *x2 = (-b – z)/(2*a) ;
        }
}
```



Summary of Lecture 3

- Invention of Programming Language
- C-Like Programming Languages
- Use of Programming Languages



Summary of Module 2

- Lecture 1:
  - Use of Natural Language

- Lecture 2:
  - Use of Technical Language

- Lecture 3:
  - Use of Programming Language

"Ask not what your country can do for you – ask what you can do for your country," - John F. Kennedy

"Do not think that you are needy – think that you are needed in the world", - Manis Friedman

"Study will make you knowledgeable, resourceful, and hence more needed", - Xie Ming

Thank You for Listening!