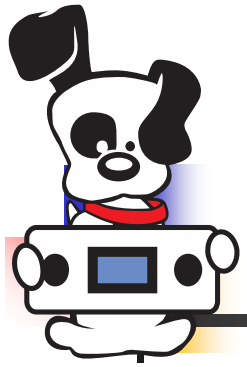


KodeKLIX for PUP

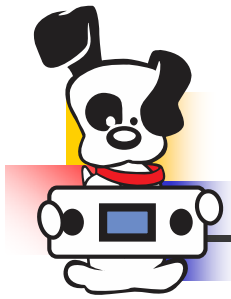
Maths, Numbers and Logic

PICAXE Maths and Numbers

- Maths for PUP programming
- Introducing INTEGER mathematics
- PICAXE numerical operators
- PICAXE variables
- PICAXE processing and limitations
- Comparative logic
- Different number systems
  - Binary, hexadecimal, character
- PICAXE logical operations

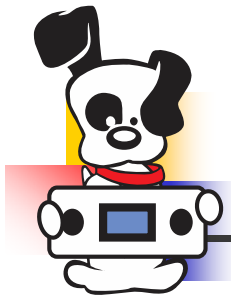
Maths for PUP Programming

- Mostly, the maths for PUP game application programming is simplified to:
  - Deciding is player actions are requested
  - Keeping track of player position
  - Deciding if player or enemy is ‘hit’
  - Updating player scores
- Advanced operations may be needed for:
  - Generating the play field
  - Creating enemy “artificial intelligence”



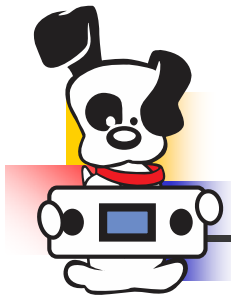
INTEGER Mathematics

- INTEGER number have no fraction or decimal part, they are whole numbers
  - Multiplying whole numbers gives a bigger whole number, eg:
    - 5 x 3 = 15
  - Dividing whole numbers gives the lowest whole number, eg:
    - 5 / 3 = 1



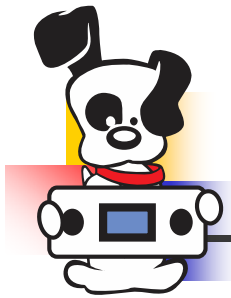
PICAXE Numerical Operators

- Common operations
  - + is add
  - - is subtract
  - * is multiple
  - / is divide
  - (...) can group operations
- For advanced operations see the PICAXE documentation or *KodeKLIX Helper*



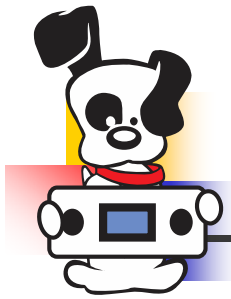
PICAXE variables

- Explained earlier in Chapter 3.0
- Example:
  - SYMBOL A = b0 ‘ from now on use A
  - SYMBOL B = 8 ‘ from now on B is fixed
  - A = 10 ‘ A is given a value of 10
  - A=A+B ‘ A now has value of 18
  - LET A=A / 2 ‘ A now is 9, LET is optional
- w0 = b1:b0 = b1 * 256 + b0



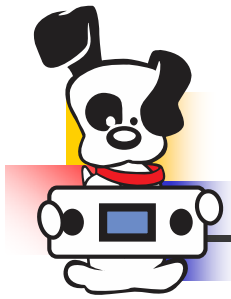
PICAXE Processing and Limits

- Most PICAXE chips process left to right in the order they encounter the operations, taking into account integer limits, eg:
  - 3 + 5 / 2 = 4 (not 5.5 or even 5)
    - 3 + 5 = 8
    - 8 / 2 = 4
- PICAXE chips have two types of numbers
  - Byte: values 0 to 255
  - Word: values 0 to 65535
  - Note, neither type includes negative values



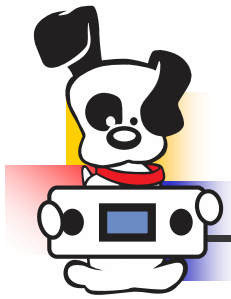
Comparative Logic

- Used in statements where a decision needs to be taken
- Common operations
  - = equal to
  - > greater than
  - < less than
  - <> not equal to (greater and less than)
  - (...) can group operations
- For advanced operations see the PICAXE documentation or *KodeKLIX Helper*



Different Number Systems

- Decimal numbers are the ones we use every day: 0, 1, 2, ... 9, 10, ...
- Computers think in binary; zeros and ones
  - 00000101 is equal to 5
  - To tell BASIC that 00000101 is 5 not 101 we use the % symbol, %00000101
- Instead of 8 digits, we can shorten the computer speak to 2 digits by grouping four 0 and 1s into a single digit, 0 to F
  - 11 = %1011 = $B



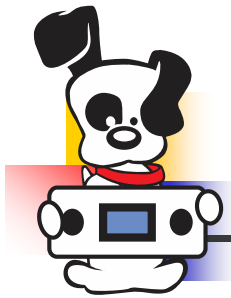
Binary Number Systems

Decimal

- 00
- 01
- 02
- 09
- 10
- 11
- 12
- 13
- 14
- 15
- 100
- 144
- 191
- 255

Binary

- %0000
- %0001
- %0010
- %1001
- %1010
- %1011
- %1100
- %1101
- %1110
- %1111
- %01100100
- %10010000
- %11000001
- %11111111



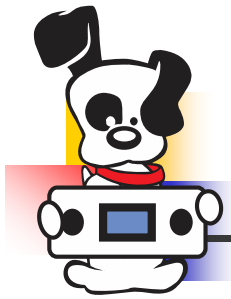
Hexadecimal Number Systems

Decimal

- 00
- 01
- 02
- 09
- 10
- 11
- 12
- 13
- 14
- 15
- 100
- 144
- 191
- 255

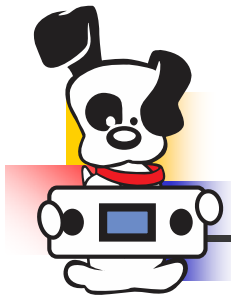
Hexadecimal

- $00
- $01
- $02
- $09
- $0A
- $0B
- $0C
- $0D
- $0E
- $0F
- $64
- $90
- $C1
- $FF



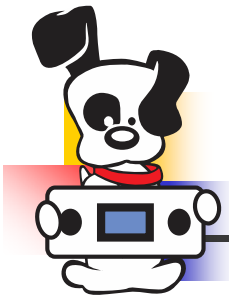
Character System

- Every character is really a number, since after all computers only understand numbers
  - ‘A’ is actually 65, or $41 or %01000001
- Character values are defined as ASCII
- Characters enclosed in single quote marks generally can be used where numbers are needed, and usually vice-verse
- Example:
  - LOOKUP b0, (01, $04, ‘A’, %0101), b1



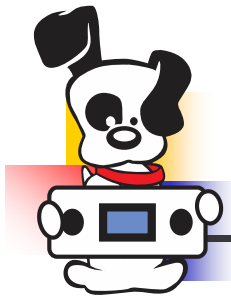
Conversions

- Automatic conversions between number systems will occur instatements and mathematic functions
- Conversion to text requires a special function, eg BINtoASCII
  - BINtoASCII b1,b2,b3,b4
    (converts b1 into text with three digits)
  - BINtoASCII w0,b2,b3,b4,b5,b6
    (converts w0 into text with five digits)



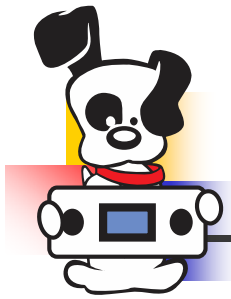
PICAXE Logical Operations #1

- Used in decision statements or in binary operations
- Common decision operations
  - AND requires multiple things to be true
  - OR requires only one thing to be true
- Examples:
  - IF A=1 AND B=0 THEN
  - IF A=1 OR B>1 THEN
- For advanced operations see the PICAXE documentation or *KodeKLIX Helper*



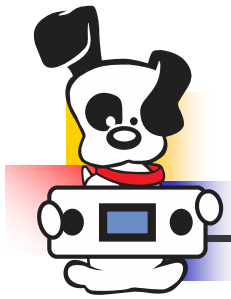
PICAXE Logical Operations #2

- Common binary operations
  - & logical AND, means keep matching 1 bits
  - | logical OR, means keep all 1 bits
  - ^ logical XOR, means invert all 1 bits
- Examples:
  - Logical AND: %0101 & %0011 = %0001
  - Logical OR: %0101 & %0110 = %0111
  - Logical XOR: %0101 ^ %1111 = %1010



RANDOMising Actions #1

- Game play is better if actions are not always sequential, eg enemies ‘randomly’ appear or the player field is ‘new’
- The RANDOM function produces a ‘pseudo’ random number
  - As its not a truly random, the function needs to be ‘seeded’ to give a different sequence each time a game is played
  - Option 1: use the TIME function
  - Option 2: call repeatedly in a key wait loop

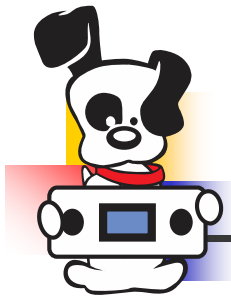


RANDOMising Actions #2

- RANDOM requires a word variable, but most game decisions don't need 65536 options, hence simplify:
  - RANDOM w0
  - b1 = b0 & 15 ‘ use lower 4 bits
  - b0 = b0 & 240 / 16 ‘ use upper 4 bits
- The resultant b0 and b1 can be used in LOOKUP or IF..THEN statements
  - Or directly as X-Y player map coordinates…

Further Help

- For advanced operations see
  - the PICAXE documentation; or
  - *KodeKLIX Helper*
- The PICAXE X series chips have more advanced math and logic functions than the simpler M series
  - M and M2 series have equivalent functions
  - X series include SIN, COS, etc
- None of the PICAXE chips floating point maths or text string capabilities



Tutorial: 3.4