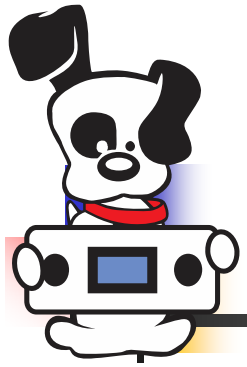
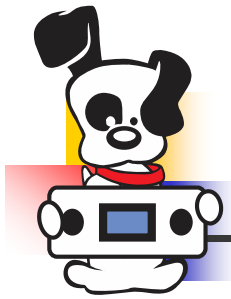


KodeKLIX for PUP

Map Based Gameplay

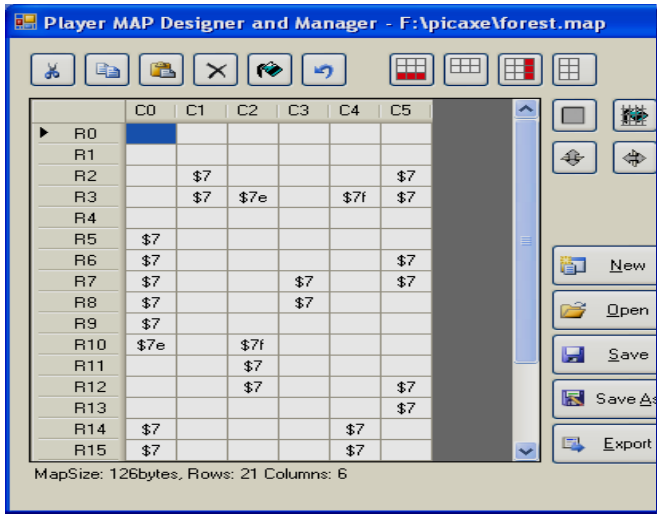


Map Based Gameplay

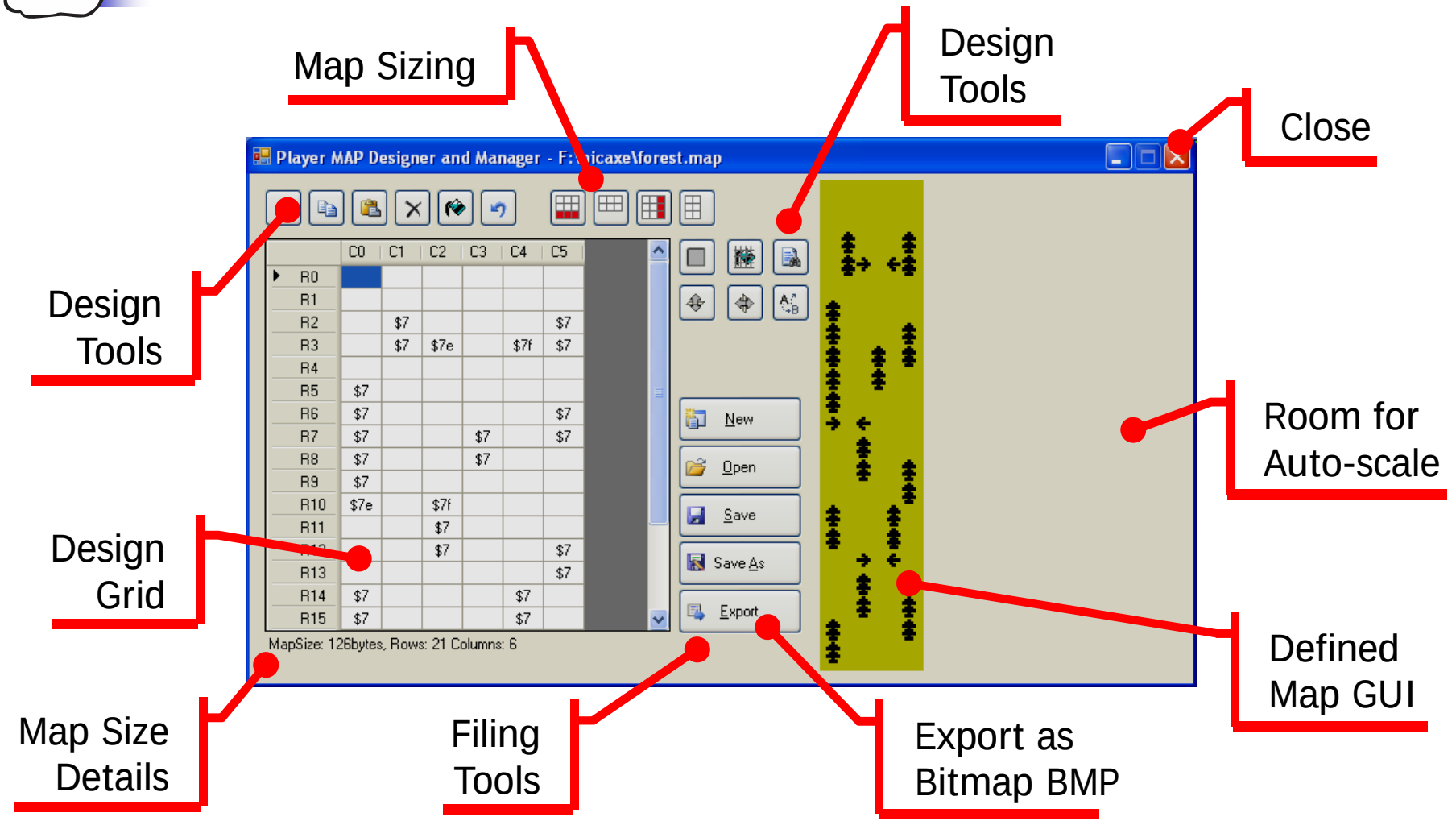
- Player Map Design Module
- Using memory to store the maps
- Code library support functions

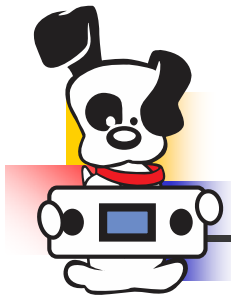
Player MAP Design Module

- Design new Player Maps
  - In games, player maps are the landscape your action takes place on
  - Note: not all games have or need maps
- Define size and aspect ratio of your map
- Place standard and custom characters
- Save Map Design
  - Share these with other KodeKLIX users



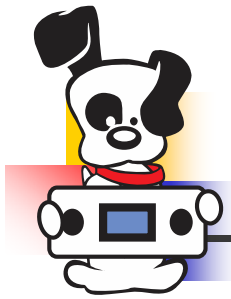
Player MAP – Grid Designer





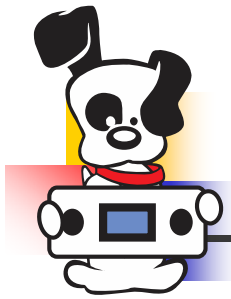
EEPROM vs TABLE Memory

- Either can be used (provided the PICAXE has it), but there are some differences
- EEPROM is writeable, therefore
  - Can be modified during game play
  - Can be randomly changed for each level
- TABLE is set at the coding stage, therefore
  - Cannot be modified at runtime
- Compromise may be to transfer from TABLE to EEPROM space at each level-up



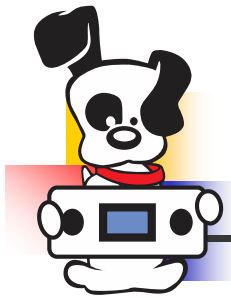
Player MAP Library Routines

- Generate Player Game Map
  - Can be randomly changed each level
- Get information from the map
  - Such as location of walls or collectables
- Collect a ‘collectable’ item from the map
  - Such as coin, treat, tools, etc.
- Set delete/change location if item can be interacted with (eg collected, destroyed)



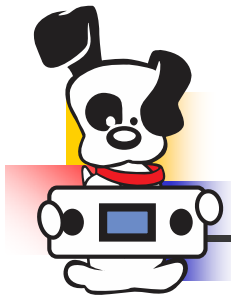
Generating a Player Map

- The routine starts by clearing the map before re-populating it.
- Player maps consist of a fixed and customisable components. You don't need to have both:
  - a player map can also be completely fixed, or
  - completely customisable.
- The fixed map is defined beforehand using the MAP tool and the data is stored on the chip, in EEPROM / TABLE.
  - The fixed map for example may define the 'background'. The macro #PlayerMap defines the drawn map.
- The customisable map is defined using the RANDOM function and information based on the game model to add some unique character to the game. The custom map may for example define the location of 'treasures' or 'targets'.



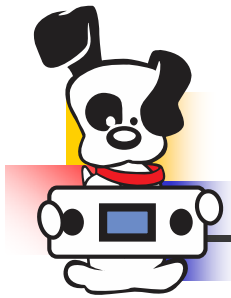
Get Information on Player Map

- These subroutines return information about the ‘item’ located in the direction that the player is moving towards
  - The ‘item’ is a character or customised character
  - The character is provided as a 'code'
- Based on the 'code' returned the app can decide to
  - Move, collect it, or do nothing about it
  - Play a tune or sound effect
  - Update the score or lives parameters



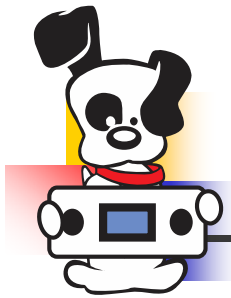
Collection from a Player Map

- Some items in a map game have significance
  - for example they may be a reward and increase your score, or a tool /key to unlock another stage.
- The 'CollectIt' routine is where the app will:
  - claim the item (clear it from the map);
  - adjust the player score; and
  - provide any visual or audio feedback to the player.



Delete Location on Player Map

- If the player's move will remove or change part of the defined map, then this set of routines will give the location to be altered.
- This is not the same as CollectIt as the player may not be moving to this location. For example the player may have 'zapped' an ‘item’, and that item now changes shape.



Tutorial: 3.8