Step by Step Instructions for Programming

These Instructions combine an introduction to programming with an outline of the historic development of the first computer games.

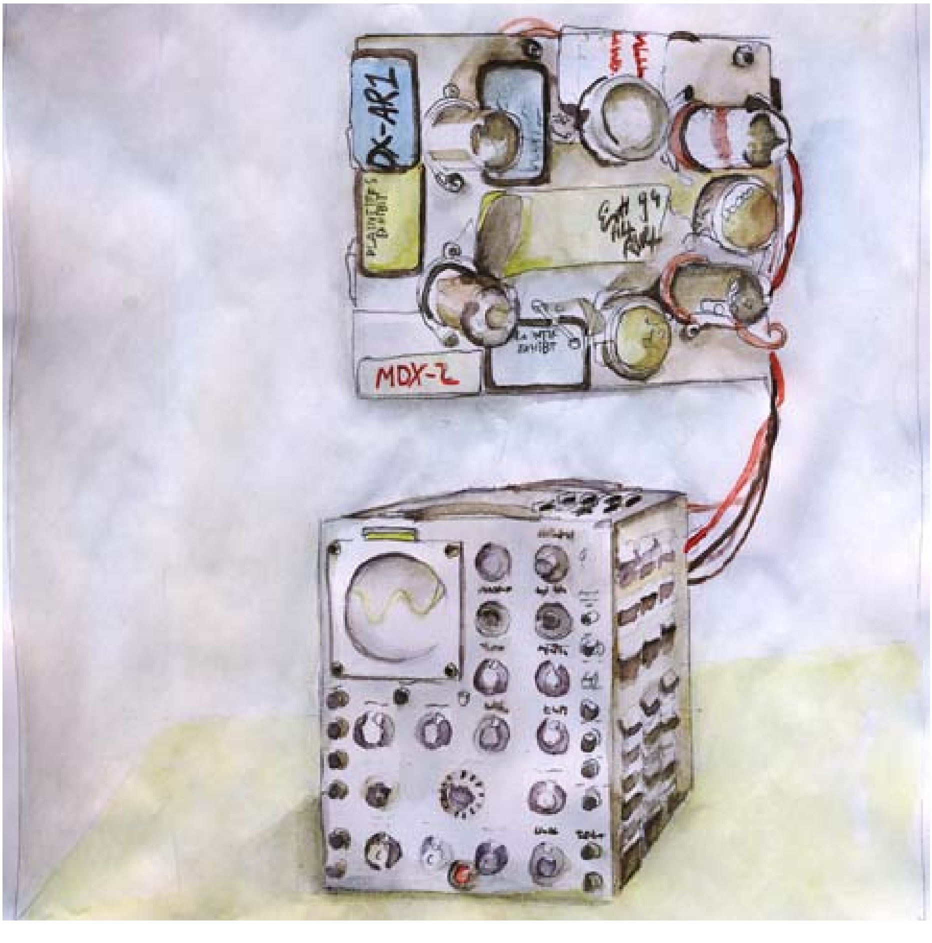
All printed programming codes can be found on the Mignon Game Kit homepage (www.mignongamekit.com) under “How-To” for download. Beginners are advised to hand- copy small programmes to thus gain access to the working process of programming.

2006 Ralph Baer (left) receives the National Medal of Technology from president George W. Bush (right)

Background History of Computer and Games

Apart from realising military applications games were also developed on the very first computers. Computers did not yet feature screens and keyboards and the developed games had very easy structures such as the “Nim Game” or the XOX Game” on the EDSAC (1949 UK). Twelve years later, after the invention of the transistor and microchips, more complex games such as “Space War” emerged. But these could only be played in the few laboratories in which computers were available. The situation changed when the television ingeneer Ralph Baer had the idea to sell inexpensive Video Games to private households in 1966.

At the time televisions generally had an inbuilt test image generator. While working on such a machine Bear observed how facinating the playful manipulation of the image through rotary switches was. His founding idea was to built a device that anyone could connect to their television. This device would draw simple synthetic forms on the display through analogue circuits- i.e. without a computer. From this games developed.



Exercise 1: “Creation of a synthetic audio signal”

In essence our first homemade programme consists of the basic structure setup and loop. Programming means writing commands that the computer can recognise underneath oneanother. When starting the programme the commands are executed from top to bottom. Firstly, there is an area for the presetting that is only exectuted once per start-up. After that a (generally endless) loop is processed. Both areas are usually framed by curly brackets.

• “File/New” is the start of a new Arduio file (Skatch)

• A commentary is the term for something that is supposed to happen. To mark such lines as commentary they begin with two forward slashes “// ”

• The connection for the speaker is to be defined as the output.

• At the exit the power is constantly turned on and off.

• We built the “Setup” and “Loop” structure:

```
void setup(){
// set sound port as output
}

void loop(){
// turn the power on and off
}
```

• Please note the differentiation between the round and curly brackets.

• Now we still need the appropriate commands to implement what is described in the commentaries.

• pinMode (number of connection, OUTPUT or INPUT);

• digitalWrite (number of connection, HIGH or LOW)

```
void setup(){
  // set sound port as output
  pinMode(9, OUTPUT);
}

void loop(){

  // turn the power on and off
  digitalWrite(9, HIGH);
  delay(1);
  digitalWrite(9, LOW);
  delay(1);
}
```

• Commands always end with a semikolon “;”

• Spacing and word wrap do not effect the functioning, but are important for the readability of the codes. The indents in particular keep interlockings lucid.

The first video game consisted of a dot placed on the screen. By use of a riffle endowed with a photocell the dot on the television could be “shot”.

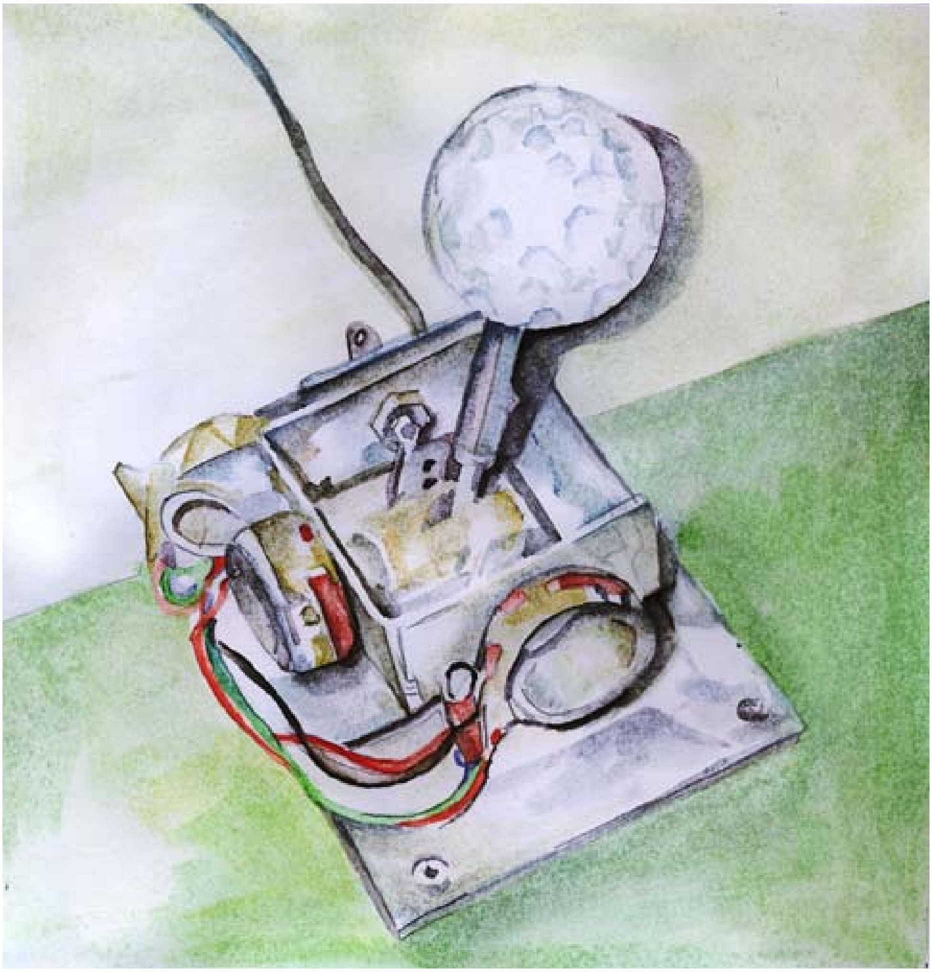
Excercise 2: “A Dot”

We put a dot on the display of the game kit. To do so we use the command „gamekit.set_pixel(row, collumn, value)“. As this command comes from the Game Kit Library, we need to first integrate and activate it. This happens with the following two programming rows: “#include ‹gamekit.h›” and “gamekit. Begin();”

```
//One Dot

#include <gamekit.h>

void setup(){
  gamekit.Begin();
}

void loop(){
  gamekit.set_pixel(2,3,15);
}
```

As we are missing the rifle sensitive to light we have to cease modelling the first video game at this point. But if one carries on putting points on the display it is possible to create signs e.g. Smileys with this little programme.

Ralph Baer continued his experiment “video game”. The next step involved moving the dot with two rotary switches and he produced first sketches of a Joystick for controlling the dots.



Excersise 3: “Move A Dot”

To move the dot the keys are queried with the following function:

```
if(gamekit.button_pressed(Konstante)){

}
```

When a key is operated the commands in between the curly brackets are executed.

In order to make the dot moveable a variable is added to the „gamekit.set_pixel()“ function for the position of the dot. These variables need to be defined with the command int (Integer) before setup. The value of these variables are lowered or raised by one with the orders “++” and “- -“. Thus the dot moves across the display. Lowering the brightness value from 15 to 0 erases the old dot.

```
// Move one Dot

#include <gamekit.h>

int row = 2;
int column = 3;

void setup(){
  gamekit.Begin();

}

void loop(){

  gamekit.set_pixel(row,column,15);

  if(gamekit.button_pressed(butt_UP)){
    gamekit.set_pixel(row,column,0);
    row--;
  }

  if(gamekit.button_pressed(butt_DOWN)){
    gamekit.set_pixel(row,column,0);
    row++;
  }
  if(gamekit.button_pressed(butt_LEFT)){
    gamekit.set_pixel(row,column,0);
    column--;
  }

  if(gamekit.button_pressed(butt_RIGHT)){
    gamekit.set_pixel(row,column,0);
    column++;
  }

}
```

This “Move One Dot” programme can easily create a “paint programme” by removing the erasure of the dots from the button query. In a further step the putting and erasing of dots can be coupled to the functioning keys.

The historic development now reached the collision query and thus a very substantial element of computer games. Ralph Baer expanded his circuits in a way that generated two dots. A second player could control the second dot. Now it was possible to play “catch”.

Exercise 4: “Chasing Game“

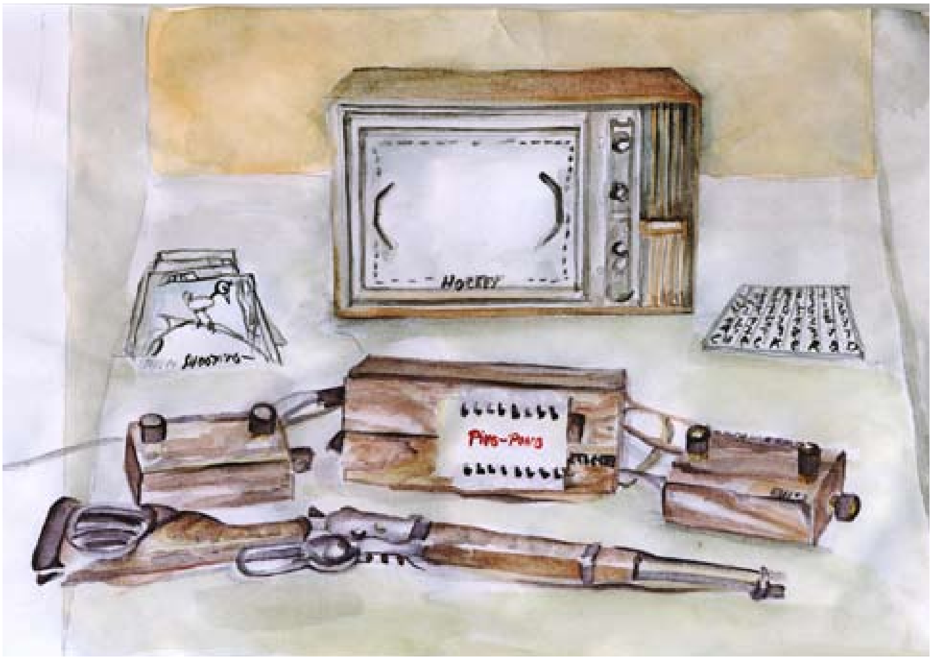
As the Game Kit only disposes of six sensors the second player runs diagonally in the chasing game. For the recognition of the collision we use an “if” query. Please note: to compare the two dots position the double equal sign is used. The single equal sign is used to allocate values to the variables. So, we differentiate between an equal sign for allocation and an equal sign for comparison. The comparisons of the “if”- query are set in round brackets. In this case two conditions need to be true at the same time. They are connected by “&&”.

To move two dots on the display the variables for the second dot are doubled (row2, column2). And the second dot is allocated the blink value 18 in order to differentiate between the two dots.

For the collision the blink value is set to 20. The “delay()” function pauses the game for two seconds. After that the dots return to their starting positions.

```
// Chasing Game

#include <gamekit.h>

int row1 = 1; // Dot1
int column1 = 1;

int row2 = 3; // Dot2
int column2 = 5;

void setup(){
  gamekit.Begin();

}

void loop(){

  gamekit.set_pixel(row1,column1,15);
  gamekit.set_pixel(row2,column2,18);


  if(gamekit.button_pressed(butt_UP)){
    gamekit.set_pixel(row1,column1,0);
    row1--;
  }
  if(gamekit.button_pressed(butt_DOWN)){
    gamekit.set_pixel(row1,column1,0);
    row1++;
  }

  if(gamekit.button_pressed(butt_LEFT)){
    gamekit.set_pixel(row1,column1,0);
    column1--;
  }

  if(gamekit.button_pressed(butt_RIGHT)){
    gamekit.set_pixel(row1,column1,0);
    column1++;
  }

  if(gamekit.button_pressed(butt_FUNCA)){
    gamekit.set_pixel(row2,column2,0);
    column2--;
    row2--;
  }

  if(gamekit.button_pressed(butt_FUNCB)){
    gamekit.set_pixel(row2,column2,0);
    column2++;
    row2++;
  }

//collision detection
  if((row1 == row2)&&(column1==column2)){

    // blink for 2 seconds
    gamekit.set_pixel(row1,column1,20);
    delay(2000);

    // go back to start positions
    gamekit.set_pixel(row1,column1, 0);
    row1 = 1;
    column1 = 1;
    row2 = 3;
    column2 = 5;
  }
}
```

The casing game gives a first impression of the possibilities of a videogame, but even during pioneering times Ralph Baer and his co-workers realised, that the game was not entertaining enough to promise great marketing success. Shortly after, the breakthrough was brought on by a special circuit that enabled the presentation of a dot on the screen that moved without being moved by a player: the ball!



Exercise 5: “Mignon Pong”

We add a third dot and built the movement of the first two dots in a way that makes them move up and down on the left and the right like pong rackets. The magical programming row that takes care of the independent movement of the ball is very simple:

```
column3 = column3 + columnM;
```

The variable “column3” is allocated a value that corresponds to its current value plus the value of the variable “columnM” in this case +1 or -1. At every call of this function the ball moves one dot to the right or to the left.

To ensure the ball moves nicely and slowly we write the entire control of the ball in an if-loop that queries the system counter of the game kit and only moves the ball at every hundreds rise of the counter. We need another variable “balltime” in which the current value of the counter is saved once, and than compared with the consecutive counter until it has moved on one hundred steps.

If the collision query is changed in such a way that the ball „hits“ the rackets and in doing so the value for the horizontal movement “columnM“ changes its sign the game almost looks like Pong. The ball moves from racket to racket but the vertical movement of the ball is missing. For this we use – for simplicities sake- random values, that are generated by the command “random()”. At the same time another if- query becomes necessary, that prevents the ball leaving the display on the top or bottom. Here again by the changing of the sign of the variable for movement “rowM” a rebound is generated.

To prevent the same random values to be generated every time the game kit is switched on the beginning of the series of random values, the so called “randomSeed”, is set to a changing value in the “Setup”. This value is best measured at the analogue input of the extension port. If no sensors such as photocells, microphone, or pressure sensor, are connected a noise is measured that delivers good coincidental values.

```
// Mignon Pong

#include <gamekit.h>

int row1 = 2; // Dot1
int column1 = 0;

int row2 = 2; // Dot2
int column2 = 6;

int row3 = 2; // Dot 3
int column3 = 3;
int rowM = 1;
int columnM = 1;
int balltime = 0;

void setup(){
  gamekit.Begin();
  //set the random seed to a noise value which is measured on the
analogue input pin 5 of the extensions port
  randomSeed(analogRead(5));
}
```

```
void loop(){

  gamekit.set_pixel(row1,column1,15);
  gamekit.set_pixel(row2,column2,15);
  gamekit.set_pixel(row3,column3,15);


  if(gamekit.button_pressed(butt_UP)){
    gamekit.set_pixel(row1,column1,0);
    row1--;
  }

  if(gamekit.button_pressed(butt_DOWN)){
    gamekit.set_pixel(row1,column1,0);
    row1++;
  }

  if(gamekit.button_pressed(butt_FUNCA)){
    gamekit.set_pixel(row2,column2,0);
    row2--;
  }

  if(gamekit.button_pressed(butt_FUNCB)){
    gamekit.set_pixel(row2,column2,0);
    row2++;
  }

  // move ball
  if(gamekit.get_systemcounter()> 100+balltime){
    balltime = gamekit.get_systemcounter();

    // collision detection 1
    if(((row1 == row3+rowM)&&(column1==column3+columnM))||(
(row2 == row3+rowM)&&(column2==column3+columnM))){
      columnM = columnM*-1;
      rowM = random(3)-1; // find a new angle for the balls path by
chance
      if(row3==4) rowM = random(2)-1;
      if(row3==0) rowM = random(2);
    }

    gamekit.set_pixel(row3, column3, 0); // turn the old dot off
```

```
   column3 = column3 + columnM; // move the dot further

   // bounce at top and bottom
   row3 = row3 + rowM;
   if(row3 +rowM <= -1){
     rowM = 1;
   }
   if(row3 +rowM >= 5){
     rowM = -1;
   }
  }
}
```

Now we reach a point at which the elementary principles of digital games have been worked through. With the learned functions new variations can be invented and games can be programmed easily.

At this stage of development Ralph Baer tried to market his product unsuccessfully in 1967. Only years later, in 1972 he was able to convince the company Magnavox to realize his vision with the “Magnavox Odyssey”. The first video console of the world was based on simple functions: Dots could be moved across the screen, moved by themselves, and it was noticeable when they hit each other. There were no sounds, no images, and no counter. Thus the “Magnavox Odyssey” included foils -with printed on motives of games that were stuck on the screen- and cards for counting scores.

So far, our exercises lack these components that turn an abstract game principle into an attractive videogame. In regard to this various possibilities exist that were discovered step-by-step in the history of computer games and lead to great commercial success. “Space Invaders” introduced the concept of several lives of the player in 1978. A figure in a game was given a name for the first time with “Pac-Man” in 1980, and the first character was established with “Mario” in 1981.

6. Exercise: „Intro Image“

We design a series of images with which we can represent small stories that are suitable to combine an abstract structure of a game with a motive. Images, sounds, and melodies raise the level of entertainment: There is identification with the avatar, an increase of sensuous appeal, and fantasy is stimulated.

To draw an image on the display we can work with the already used “set_pixel” function. However, the “load_image” function from the Game Kit Library is more comfortable.

```
uint8_t myman[5][7] PROGMEM = {
// Dot Values
};

gamekit.load_image(myman);
```

As the images use a special type of variables the library “pgmspace” needs to be integrated. After that the images are immediately defined with the variables. This is a task for the advanced: One uses an array of Unsinde Integer 8 bit variables that are placed in the programme memory. You do not have to deal with that right now! The easiest way is to insert the respective codes using “copy and paste”. Instead of “myman” any chosen title can be used for the image. The five rows of numbers with the seven values represent the rows of the display with their seven LEDs. At the value of zero the diode is turned off. At the maximum value of 15 it shines the brightest. The values in between allow for working withnuances. The “load_imge” function loads the image onto the display.

```
// Intro Image

#include <gamekit.h>
#include <avr/pgmspace.h>

uint8_t myman[5][7] PROGMEM = {
  0 ,0 ,0 ,15,0 ,0 ,0 ,
  0 ,3 ,3 ,3 ,3 ,3 ,0 ,
  0 ,0 ,0 ,3 ,0 ,0 ,0 ,
  0 ,0 ,3 ,0 ,3 ,0 ,0 ,
  0 ,0 ,3 ,0 ,3 ,0 ,0 ,
};

void setup(){
  gamekit.Begin()
}

void loop(){
  gamekit.load_image(myman);
}
```

7. Exercise: „Intro Animation“

You can easily produce an animation by simply defining several images and open them one after another. The speed of the changing of the pictures is defined by the “delay” commands behind the “load_image” commands.

The following example follows a more elegant path. Again only one image is defined. However, it functions like a film reel. In the example the dancing man consists of a series of five images. This film reel is pushed from the bottom to the top across the display. Thus an animation is generated.

To load an image that is bigger than the 5 x 7 grid you use the “load_map” command. In this case the image is moved in its entirety, i.e. it jumps up 5 rows. For the line- pulling the variable “i” is used, that is raised with the command „i+=5“ at each round. An „if“ command returns „i“ to zero.

```
// Intro Animation

#include <gamekit.h>
#include <avr/pgmspace.h>
```

```
uint8_t dancingman [25][7] PROGMEM = {
  0 ,0 ,0 ,9 ,0 ,9 ,0 ,
  0 ,9 ,9 ,9 ,9 ,0 ,0 ,
  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,0 ,9 ,0 ,9 ,0 ,0 ,
  0 ,9 ,0 ,0 ,9 ,0 ,0 ,

  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,9 ,9 ,9 ,9 ,9 ,0 ,
  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,0 ,9 ,0 ,9 ,0 ,0 ,
  0 ,0 ,9 ,0 ,9 ,0 ,0 ,

  0 ,9 ,0 ,9 ,0 ,9 ,0 ,
  0 ,0 ,9 ,9 ,9 ,0 ,0 ,
  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,0 ,9 ,0 ,9 ,0 ,0 ,
  0 ,0 ,9 ,0 ,9 ,0 ,0 ,

  0 ,0 ,0 ,0 ,0 ,0 ,0 ,
  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,9 ,9 ,9 ,9 ,9 ,0 ,
  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,9 ,9 ,0 ,9 ,9 ,0 ,

  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,9 ,9 ,9 ,9 ,9 ,0 ,
  0 ,0 ,0 ,9 ,0 ,0 ,0 ,
  0 ,9 ,9 ,0 ,9 ,0 ,0 ,
  0 ,0 ,0 ,0 ,9 ,0 ,0 ,

};

void setup(){
  gamekit.Begin();
}

int i = 0;

void loop(){

  gamekit.load_map( (uint8_t *) dancingman, 25, 7, i, 0);
  delay(600);
  i=i+5;

  if( i >= 25 )
    i = 0;
}
```