Micro:bit + Processing

shooting game

Lazăr Cătălin-Mihai

Vaida Diana-Laura

Group 3

The project consists of a game similar to Space Invaders where the player has to dodge falling objects and can also shoot them down. It was made with the help of the Processing IDE for the interface and the micro:bit hardware embedded system for controlling the player.

**Micro:bit**

To control the player position and actions, the micro:bit board was used. The data from the micro:bit is sent to Processing via the serial interface.

More exactly, at each iteration, the x value of the acceleration (from a sensor on the board) is used to control the moving direction of the player. If the acceleration on x > 60, the board is considered to be tilted to the right, and it will send the character „E” on the serial interface. It x<-60, the board is considered to be tilted to the left, and the „W” character is sent on the serial interface. Also, if button „a” is pressed, the character „F” is sent, which means that the player will shoot a bullet. If the button „b” is pressed, the board will send the character „R”, which the processing program interprets as a „Restart” command, in case the player has 0 lives left.

Also, depending on the tilting direction (righ, left or no tilt), an image in the form of an arrow is drawn on the 5x5 led display of the micro:bit.

**Processing**

The Processing IDE is used to draw the scene on the screen. The code is written in the Java language. The game has the same number of 10 falling objects that appear again and again on the screen. Every object has a X,Y coordinate for the position and also a velocity. Adding the velocities to the position of the objects causes them to move down. The bullets are declared as an ArrayList of pVector(containing the X,Y coordinate for every bullet on the screen).

The player has a number of 3 lives initialized at first. The initial score at the start of the game is 0. The player cannot fire bullets continually but has to wait a certain number of frames (20) between firing 2 bullets. Also, after being hit, the player has a period of 60 frames in which he has immunity. The longer the player is alive, the higher the score becomes. The difficulty is increased as time passes, and the speed of the falling objects becomes faster. However, the increase of the difficulty is not constant, but it slows down over time.

The same 10 falling obstacles are reused repeatedly to save memory and speed up the game. When an object is shot or passes the player, it is again spawned above the screen.

When the game receives the character “R” from the micro:bit, the game will restart, but that can only happen when the player has no more lives. The “F” character makes the player shoot and a new bullet will be generated using the player coordinates. As the bullet goes up the screen, its Y coordinate decreases.

When the “W” (west) character is received, the player will move to the left of the screen, and when the “E” (east) character is received, the player will move to the right.

Another important mechanism is the collisions with the falling objects. If the distance between the player and the ball is less than 20, and the hit cool down counter is 0 (the player isn’t red and isn’t in a vulnerable state), then the player was hit, and loses a life, and the hit cool down counter resets again to the maximum value and can’t be hit for a period of time. There is another function used to check the collision of bullets with obstacles.

The scene, objects, player, bullets, text are drawn on the screen using specific functions in processing like ellipse(), fill(), background(), text().