**Basic IO shield resourced Space game**

**By Murat Eksi (970218-4897) and Justin Arieltan (990401-2359)**

**2018-02-28**

* **Objective and Requirements:** The purpose of this project is to make a space game that is inspired by “Space Impact” on the old Nokia platforms and optimize it well enough to run as smooth as possible on the Uno32 embedded system. We have aimed for an advanced project.

**Things implemented:**

* 1. The game supports high score lists and different difficulty levels.
  2. The orientation of the same type of objects such as enemy ships or the projectiles must be implemented.
  3. The player is able to control the main ship using the buttons for X-axis and shooting, while using the potentiometer for Y-axis.
  4. The sprites of the game move fluidly throughout the screen, pixel by pixel.
  5. Hitbox for the objects
  6. Particular things such as the health of player must be outputted in some way using the basic IO shield (led, screen and etc..)
  7. High Score is being saved

1. More advanced AI for the enemy ships
2. Menu Interface (Play, High Score, Difficulty select)

* **Solution:** We have developed our game for Uno32 and the basic IO shield of it. We have used the display on the basic IO shield to output the game for the player to interact with. The buttons on the basic IO shield have been used for X-axis movement and shooting while the potentiometer have been used for Y-axis movement. We have outputted the health and the high score to the player using the led system on the basic IO shield and also implemented the pause mode and changing of the LEDs in regard of health and score using the switches on the basic IO shield. The difficulty will be a dynamic factor that depends on the current score which results in more ships and we also made it possible to change while we are in the main menu. We have implemented our system of pixel drawing for the objects in order to print them to the screen as smooth as possible and also made it possible to have a hitbox for the assets using arrays as a data structures solution for object orientation. Refresh rate of the game and the speed of it mostly depends on a delay function we have implemented. We also had a problem with the random number generator of the standard library in C, so we implemented our own random number generator for our project using the included timer 2 in the system. We also managed to save the high score to the EEPROM using the I2C protocol. Most of our project is written using the C and it is also compiled using MCB32tools.
* **Verification:** We have tried some stress test on our EEPROM saving strategy by having edge cases of random resetting, plugging it off and restarting the Uno32. We have also played our game repeatedly to find bugs and we have fixed them like the hitbox and enemy movement. We have let our friends to play and test and comment on the stuff so that we could alter our strategy in a better way.
* **Contributions:** We have worked together on designing the game, asset design and did research in regards of implementing the game to the Uno32. Murat has done the game logic such as intro, buttons, menu, high score, difficulty, movement, object orientation, hitbox, enemy AI, LEDs and pixel by pixel drawing. Justin has done the game over animation and the I2C EEPROM has been implemented by Murat.
* **Reflections:** We believe that by doing this project, we have really learned a lot regarding the connection protocols such as SPI and I2C. We have also developed our skills on designing and implementing our games to a device that we didn’t use before and we have acquired new skills in regards of project management and team work. We also believe that, this was also a great opportunity to develop our C and assembly skills, such as pointer and data structures. We also got really comfortable in regards of reading manuals and working through them.
* **Appendix**

The code for our project can be found at:

