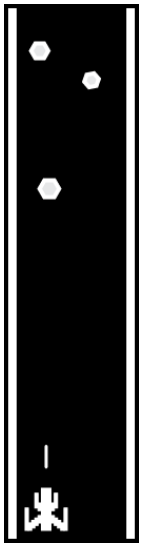
UNOSHOOTER

FINAL EXTENDED ABSTRACT

ADVANCED PROJECT



KTH Royal Institute of Technology

IS1200 Computer Hardware Engineering

Alex Diaz
alexdiaz@kth.se
970323-2184

Johan Edman
jedma@kth.se
970613-9079

Alex Diaz
Johan Edman

Table of Contents

Alex Diaz
Johan Edman

Objective and Requirements

The aim of this course project is to gain a deeper understanding of computer organization, hardware and software by developing end-user-friendly software on the *ChipKIT Uc32* platform. To emphasize the core of hardware engineering, the software is written in the C programming language.

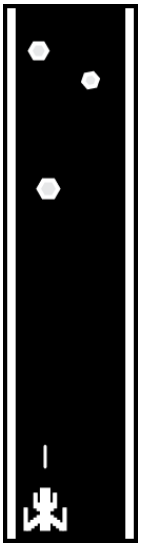


Figure 1. Concept art (4:1)

The main objective of the project is to develop a vertical 2D side-scroller clone of 'Asteroid Blaster' on the *Uc32* platform. The game receives input from the player via the onboard *ChipKIT* buttons and potentiometer. As the player flies around in space, rocks will come tumbling down, as illustrated in *Figure 1*. To survive the user must either dodge the rocks or blast them to smithereens with the laser. The measure of success is for how long one player can survive, and how many asteroids that have been destroyed, as each rock destroyed gives 5 points. If the player collides with asteroids three times, the ship will break down and the user is forever lost in the endless void that is space.

The minimal required functionality of the game is as following:

- The user must be able to control the game with the buttons and potentiometer on the *ChipKIT* board.
- The game must keep track of collisions between player and rocks as well as laser-shots and rocks.
- The game's difficulty should increase with time in a linear matter.
- The game should implement and keep a persistent high score.

Some additional features which could be added to make the game more immersive are:

- A difficulty setting which manages how many lives the player has, i.e. how many hits the player can take before falling. This could be displayed via the LEDs.
- Including various power-ups or special events that either help the player or make it more difficult.
- Use of the A/D converter together with an external speaker to generate sound effects.

Solution

The foundation of the solution to the project lies upon using the *ChipKIT Uc32* in combination with the Basic I/O shield. The display will be used to display the game itself, including the player and the game environment. The player will be able to interact with the game via three buttons and the potentiometer, allowing them to move in the X, Y direction and shoot lasers.

With the use of interrupts that are triggered by the internal timers within the unit, the state of the game will be updated to the display and control the speed. Timers will also be used to limit how often the player can shoot and grant a short period of invincibility after being hit by a rock.

The project code, which will implement the C programming language (C99 Standard), will be stored on a GitHub page allowing the members of the project to easily access and commit changes to the code. The code itself will then be compiled and flashed to the device with the *mcb32tools* compiler provided by the course.

Verification

The main idea is to have a TDD (Test-Driven-Development) workflow where the game is divided into various functional parts which can be tested and verified, hopefully, separately. The TDD functionality can for example, be automatically integrated via the GitHub repository with CI (Continual-Integration) software. This makes sure that the any refactoring or changes to code maintains its functionality and works as intended.

There will however be some exceptions to this, such as verifying code that interacts with peripheral devices connected to the board, and certain game logic itself. These exceptions/tests, will instead have to be verified manually. This should be done in a step-by-step process where, as earlier stated, small functions are tested, and their functionality verified, thus giving the "big-picture" functionality.

Contributions

The aim is to split the workload evenly, however as time passes it's certain that each member of the project will prove to be good at different things and a natural distribution of the areas involved will occur.
Update. The workload, programming of game logic and configuration of I/O has been split pretty much evenly. For further in-depth detail, one can turn to the code, which is stored on GitHub, where comments state more specifically what has been done by who.

GitHub: https://git.io/UnoShooter (http://bit.ly/UnoShooter - Only works if Public Repo)

Reflections

The project has turned out to be a stimulating experience in both a fun and challenging way. There have been many bumps on the road, things that have not always worked out exactly as planned, but given time, most problems were solved and other mitigated to the furthest extent possible.

We have certainly learned a lot about the *ChipKIT* and how it is to code outside the safe environment of an IDE, and writing "everything" from the ground up.