VIDEO 15.1 OVERVIEW OF THE REQUIREMENTS DOCUMENT

C15 1 Requirements Document

6:46 / 6:46 1.0x

DR. RAMESH YERRABALLI: So as we saw in the introduction,

this module will bring everything that you learned into one single concept.

DR. JONATHAN VALVANO: Yeah, we can build a game.

DR. RAMESH YERRABALLI: So the game that we're going to build

is the one that Jon is playing right here.

DR. JONATHAN VALVANO: And losing.

DR. RAMESH YERRABALLI: It's a space invaders game.

OK, let's give it another try, So we will collect first

the requirements for this game.

A requirements document is the part where we start.

DR. JONATHAN VALVANO: So a requirements document

is a legal contract between the developer, which is you,

and the customer, which will purchase your software.

DR. RAMESH YERRABALLI: So the requirements document

has to spell out in great detail the requirements of what the system ought

to look like, what are its features, what its form factor should be,

and what the cost ought to be, and what components it may or may not

need so that a standard can be delivered.

So now we'll look at what discussions or exchanges go

in building the requirements document.

DR. JONATHAN VALVANO: The requirements document has three parts.

The first is an overview.

Why are we doing this project?

We're doing this project to put together all the pieces of this class,

and in particular, we're going to build a 1980s style shoot 'em up

game like Space Invaders.

The second part of the overview is the process,

and the process for developing this game will be like all the other labs.

And that is we will begin with a starting point or a starter project,

and you will add features to this starter project

to meet the requirements of this lab.

DR. RAMESH YERRABALLI: So, Jon, are you saying

that they don't have to design this project from scratch?

DR. JONATHAN VALVANO: That's what that means.

The other aspect of this project, which is different than the other labs,

is since we have no grader that will automatically grade your lab,

we encourage you to find teams and work together

to perform or to develop your game in teams.

We suggest somewhere between one and three members of your team.

The most important part of the requirements document

that you want to look into is exactly how does your game fit.

In other words, there are some specific requirements that your game must have.

It must be written in C, it has to be developed on Keil,

it has to run on your LaunchPad, and more specifically, we

have decided or defined exactly where the input and output pins are located.

By specifying exactly where the pins are,

this will allow one student to play the game of another student.

When reading the requirements document, you

will see we'll use specific terminology like BMP files, sprites,

and public documents.

And you can read and read those definitions.

We are worried about you spreading viruses from one student to another.

So we are only going to allow the upload of one source code file.

And since this game is not required for your final course grade,

you don't have to actually upload it.

DR. RAMESH YERRABALLI: So you're saying that if I build a software,

I could send you my project, just one single file which has all the source

code.

And you will run it, and it will work perfectly on your system

because you built a hardware exactly to specification

that my software is following.

DR. JONATHAN VALVANO: Absolutely.

That allows us to play each other's games.

The second part of the requirements document is the functional description,

and it specifies exactly what the system will do.

It'll be an '80s style video game with inputs and outputs.

These are the phases of development .

You will find a team, if you'd like, to find

exactly what the rules of your game will be.

We'll specify the modules, the various hardware and software components.

We will organize or architect your software

by defining prototypes for the public functions and then one by one,

you will test the modules.

And then once all the modules are tested,

you will put them together to create a system.

DR. RAMESH YERRABALLI: So one of the things it looks like,

Jon, is that you're hiding all the details of implementations

by exposing only the prototypes of public functions to the other modules.

DR. JONATHAN VALVANO: This is a common development process for larger systems,

and that allows more than one person to work on the project at a time.

We'll evaluate the beauty or the wonderfulness of your game

by two very, very simple properties.

Is it easy to learn to play the game, and second, is your game fun to play?

DR. RAMESH YERRABALLI: Oh, you mean if I have

to read a document to play the game, it's not easy.

DR. JONATHAN VALVANO: It's not easy.

DR. RAMESH YERRABALLI: And if I have to turn a knob a certain way

or do seven things before I can make a missile fire, that's not fun to play.

DR. JONATHAN VALVANO: No, and we'll have an option

to allow you to share your game with other students in the class.

The usability is exactly how it is the game will occur, and as I said,

we'll have very, very, very explicit pins,

which must be for your input and your output.

And these requirements are specified in detail,

and it will allow one student to play the game of another student.

And again, because we're worried about spreading viruses,

we will only take one C file of all your programs in text form.

And the last part of a requirements document is the deliverables.

We won't make you write a report, but we do

encourage you to put comments in your C file.

We don't have any audits, but we will allow

you to talk about your game in the forums,

and the ultimate deliverable, like we said,

will be one large C file that we can upload and share with the class.

All right, let's start building.

Help

Back in Chapter 7 we presented an outline of a **Requirements Document**. A requirements document states what the system will do. It does not state how the system will do it. The main purpose of a requirements document is to serve as an agreement between you and your clients describing what the system will do. This agreement can become a legally binding contract. We should write the document so that it is easy to read and understand by others. It should be unambiguous, complete, verifiable, and modifiable. In this chapter we will use the framework of the requirements document to describe the hand-held game project.

1. Overview

1.1. Objectives: Why are we doing this project? What is the purpose? The overall objective of this project is to integrate the individual components taught in this class into a single system. More specifically, the objectives of this project are: 1) design, test, and debug a large C program; 2) to review I/O interfacing techniques used in this class; and 3) to design a system that performs a useful task. In particular we will design an 80's-style shoot-em up game like **Space Invaders**.

1.2. Process: How will the project be developed? Similar to the labs, this project will have a starter project, Lab15_SpaceInvaders, which will include some art and sounds to get you started.

1.3. Roles and Responsibilities: Who will do what? Who are the clients? Students may develop their games individually or in teams. An effective team size for this project ranges from 1 to 3 members. There is no upper limit to team size, but above 3 members will present difficulty in communication and decision making. The clients for this project will be other classmates and your professors of the UT.6.01x course.

1.4. Interactions with Existing Systems: How will it fit in? The game must be developed in C on the Keil IDE and run on a Stellaris/Tiva LaunchPad. We expect you to combine your solutions to Lab 8 (switches, LED), Lab 12 (interrupts), Lab 13 (DAC and sounds), and Lab 14 (slide pot and ADC) into one system. We expect everyone to use the slide pot, two switches, two LEDs, one 4-bit DAC, and the Nokia5110 LCD screen. If you do not own a Nokia5110 there will be a mechanism to

pass the Nokia graphic commands to the PC and the application TExaSdisplay will show the images in real time as your game software runs on your real LaunchPad.

1.5. Terminology: Define terms used in the document. **BMP** is a simple file format to store graphical images. A **sprite** is a virtual entity that is created, moves around the screen, and might disappear. A **public** function is one that can be called by another module. For example, since the main program is allowed to call **`Sound_Play`**, **`Sound_Play`** is a public function.

1.6. Security: How will intellectual property be managed? Since this project does not contribute to the final grade in UT.6.01x you do not need to upload your solution. If you do upload your source code, then other students who have uploaded will be able to see and download your source code. To reduce the chance of spreading viruses, we will restrict the upload to a single text-formatted source code file. More specifically, the upload must be a `SpaceInvaders.c` file, and this file must compile within a project like the `Lab15_SpaceInvaders` starter project within the 32k-limit of the free version of the Keil IDE.