Starkas

Technical Design Document

Executive Summary

This document will describe how the Starkas project, for the Nintendo DS, will be developed and what tools will be used. Starkas will be developed using C++ in the Programmer's Notepad integrated development environment (IDE), as well as Microsoft Visual C++. Testing the software will be conducted on a physical Nintendo DS as well as Nintendo DS emulators, namely DesMume. The library is Nintendo's libnds directory supported by DS Homebrew.

Equipment & Tools

First and foremost, computers are needed to develop this project. Our team has access to a laptop as well as a desktop. Internet is vital to this project, not just for support through issues we encounter, but to communicating effectively as well, and sharing our work when not in the same area. Flash drives as well will be important to sharing files quickly with each other when we are working together.

The second most important thing to programming on the Nintendo DS is to have the library as well as software supporting this library in an IDE. We have access to the libnds library, used specifically for programming on the DS, free through the DS Homebrew site. They not only offered the library, but also included Programmer's Notepad fully working with libnds and up to date emulators as well. Programmer's Notepad lacks some functionality of IDE's the team is used to, such as Intellisense and breakpoints, so being able to use Microsoft's Visual Studio with C++ will help us debug code more efficiently before incorporating it into the main project on the Programmer's Notepad IDE.

Testing the game will require specific hardware and software. First, we will ensure the project works on our computers on emulators first, of which we will mainly be using DeSmuME. If everything is working well, we will then transfer it to a physical

Nintendo DS system via an M3DS Real memory chip. This chip has a slot for a microSD and with an adapter, fits in any SD card reader. The stylus that comes with the Nintendo DS will also be used in testing the software.

Programming Language

The programming language being used, as mentioned previously, will be C++. C++ is a powerful language, and used very commonly in the game industry. Though the team is not used to programming in a memory-efficient way, it will be to our benefit, as we will be learning such techniques before getting started on our career, and in the end we will have produced an efficient game. There is plenty of support and documentation for C++ on the internet, and as such issues will be kept to a minimum. C++ also allows programmers control over how the hardware works with one another, allowing there to be no wasted processing power.

DS Hardware Talk

To code for the DS, it means that you must have some knowledge of the hardware and how it works. The two processors of the DS dedicate themselves to their own screen. An OAM and VRAM (object and video memory) is also used and shared between both processors. Direct manipulation of bits, and storing graphics requires functions calls (in libnds) to "get" the length of the graphic (in bits) to apply the next graphic afterwards in memory. Also to copy information to the OAM you must create your own copy of the OAM and copy it over to the real OAM. The reason this is done is because during draw, the OAM is locked and you must wait until the screen is in its Vblank and Hblank states before you can access it. Vblank and Hblank stands for vertical and horizontal blank scan lines which indicates that it is between drawing frames. Also, both processors can be in different modes ranging from 0 to at least 5. The mode we have both processors in is mode 5. This is because of its flexibility and compatibility with 2D, and optional 3D effects if desired. Modes can be changed and both CPU's do not need to be in the same mode.

Platform Evaluation

Starkas is being developed for the Nintendo DS, so there are some challenges related with this platform. For one, developing the functionality in the game with the stylus and the screen the stylus is used with. We will be using Homebrew's DS development kit, offered as a free download from their website.

Code Structure – C++

Variable names will make use of camel casing, whereas methods and classes will start with a capital letter, but continue the same camel casing style: int numOne; Class PlayerOne{}; etc. Comments should be included for every block of code, and summarize what that block is doing. It would also be important to do this if a few related but noteworthy lines of code can be confusing or misunderstood. Since Programmer's Notepad does not come with functionality for automatically indenting appropriately, it is important we do these ourselves consistently. A primary method, such as main(), should be aligned to the very left edge of the screen. Open and closed brackets appear underneath main(), and any code entered in there is indented one spot over. Every method and class will follow this approach, where, basically if there is code entered in a block contained by brackets; it should be indented one spot to the right of where the brackets are. If a method call or class constructor has enough parameters to where it trails off the visible edge of the screen, after the comma in one of the parameter passes, a new line should be created and indented to where the following parameters passed are roughly underneath that first parenthesis of the function call and indented once more. Comments at the beginning of blocks of code should be aligned with the line of code directly following it below.

As far as the different components, there should be a Render class made specifically for the act of loading and initializing as well as clearing the appropriate memory slots for graphics. There is also an AI component which will calculate how the enemy ships act. This will make use of a finite state machine, and, if time permits, will incorporate the use of a messaging system. For the player and enemy objects, there will be classes for each that derive off of a parent Ship class which inherits from the Entity class. These classes hold information for moving the ships, setting up collision boundaries, and rotation. Player and Enemy classes will inherit off of Ship, and define these functions in their headers and cpp files. The Bullet class will be a separate class, which can be used by both the player and enemies in the game. The class will store information about a bullet's velocity and bounding box.

Development Plan

The first thing needed for the team was to download the proper software used for coding and testing the game. Then, ensure we have access to a physical Nintendo DS and stylus for final testing purposes. Currently, the team has had all the necessary tools for developing on the DS. As a work schedule goes, we are sticking to a minimum 10 hours a week regime. We meet once per week aside from class to see what needs to be done, work on development together, and agree on assigning small tasks to each member to complete over the course of the week. First, we will be working on functionality. Once that has been accomplished, we will work on ensuring that the game has the right graphics, as well as incorporate sound effects for the finished product.