Use of Ada in Digital Radar Landmass Simulation (DRLMS)

Jim Hopper
Science Applications International Corp
4031 Colonel Glen Hyw
Dayton, OH 45432
937-431-2336
Jim_hopper@dayton.saic.com

Harry Heaton
Science Applications International Corp
4031 Colonel Glen Hyw
Dayton, OH 45432
937-431-2324
Harry_heaton@dayton.saic.com

Jennifer DeVilbiss
Science Applications International Corp
4031 Colonel Glen Hyw
Dayton, OH 45432
937-431-2325
jennifer_devilbiss@dayton.saic.com

Tom Haberlandt
Science Applications International Corp
4031 Colonel Glen Hyw
Dayton, OH 45432
937-431-2326
Tom_haberlandt@dayton.saic.com

1. ABSTRACT

In this paper we describe our experiences in the use of Ada in the development of SAIC's Digital Radar Landmass Simulation known as Radsim.

1.1 Keywords

Ada 95, Object Oriented Development, Real Time,Simulation.

2. Introduction

Radsim(TM) is a line of cost-effective, high fidelity, multi-mode radar simulations implemented entirely in software. This approach allows Radsim(TM) to easily adapt to a wide variety of training system applications. The latest version consists of SAIC's patented core radar simulation technology, including the fundamental algorithms and processes necessary to produce real-time synthetic aperture radar, doppler beam sharpening and real beam imagery. Implemented as an Ada95 class library that allows the construction of new Radar simulations with a fraction of the work required to build custom simulations. Figure 1 is a Radsim generated 1 meter Synthetic Aperture Radar image of some buildings. Note the highlights and shadows for individual buildings.

3. Initial Work

RADSIM(TM) is based on a highly successful approach developed for simulating the AN/APQ-164 radar on the B-1B Engineering Research Simulator built by SAIC for the Armstrong Aerospace Research Laboratory at Wright-Patterson Air Force base in 1984. One of the driving requirements of the B-1B DRLMS was a high degree of simulated image fidelity and correlation with real-world radar imagery. This original DRLMS was developed in a combination of PLM and FORTRAN. It was highly successful, but extremely expensive to maintain and tied to SAIC proprietary hardware.

A prototype, software-only solution was done on the Apple Macintosh to explore the feasibility of a software only solution about 1987. This prototype also served as a testbed for developing new algorithms to improve the fidelity of the Radar simulation. This prototype, which simulated the B-1B Synthetic Aperture Radar in software on an Apple Macintosh IIcx computer (Motorola 68030 processor), was highly successful and used an astonishingly small amount of computer resources for its time. The prototype had roughly the same performance as the original hardware simulation, with vastly improved fidelity and flexibility. This work was done in Pascal and was highly tied to the Apple MacOS graphics libraries. The budget for this development was all but non-existent, and a quick, cheap implementation was the overwhelming priority. The platform choice was driven by Apple's toolbox support for graphics, which made implementing the core image processing algorithms much quicker than on competing platforms. The language selection was driven by Pascal's dominance in Macintosh software development at that time. In the end, this prototype showed we could achieve sufficient performance to transition our simulation from its hardware support to a software solution. It was also an excellent testbed, as well as a superior marketing demonstration of the technology.

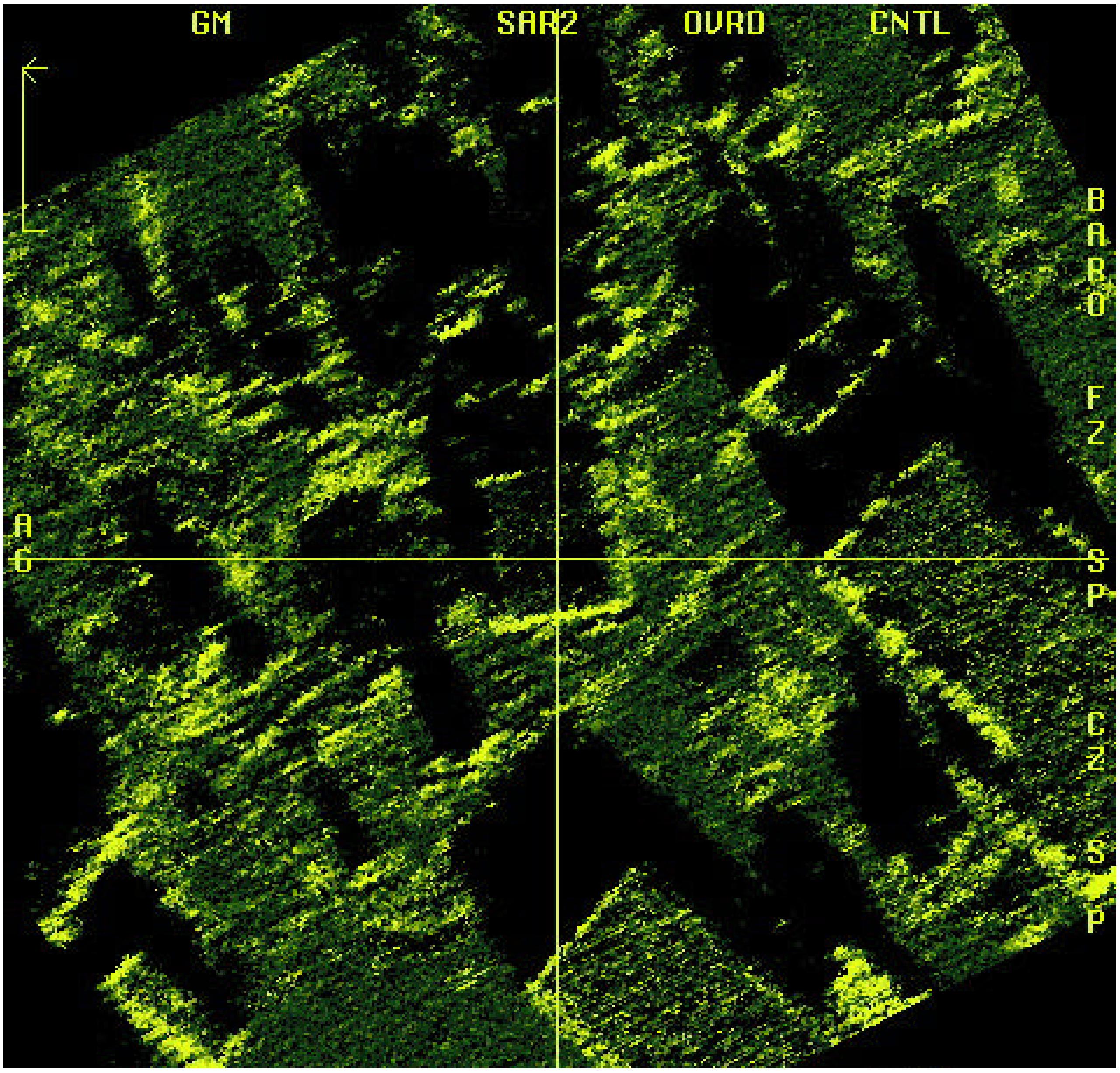
Hughes Training, Inc. (Now Raytheon), selected RADSIM(TM) as the optimal solution for meeting AN/APG-68 radar simulation requirements associated with the F-16 Unit Training Device (UTD). This was based upon its high fidelity combined with its very low cost per unit. Maintainability, reusability and portability had all became very important issues. Ada was a natural choice to meet these requirements, and the RADSIM(TM) code was ported to Rational Ada 83 on a Silicon Graphics Onyx. It was also extended to simulate F-16 specific Radar modes that had not been available in the original prototype. The UTD simulation combined the flight simulation, DRLMS, and Visuals in one SGI Onyx box. The original requirement was for the DRLMS to run on two SGI R4400 processors on-board the UTD Onyx computer, but we ultimately delivered the UTD DRLMS operating on a single 200Mhz R4400 processor, and did not use the Onyx graphics pipe at all.

The source code from this program and its test driver was ultimately ported to Ada 95 using a beta of GNAT on the SGI. No algorithms were changed and no Ada 95 specific features were used. The GNAT version executed approximately 25% faster than it was under Rational SGI Ada 83. The port was accomplished with minimal effort (about two weeks) by a developer who had minimal familiarity with the original UTD source code.

4. Radsim Redesign

Squeezing the UTD radar from 2 CPUs down to only 1 made a great deal of sense financially for the UTD program, however the optimizations required to do this forced us to compromise a number of our goals for the RADSIM software. The flexibility of the code that was to allow simulation of diverse radar systems suffered a great deal in this process as well as the rehostability of the software. When we completed the UTD project, we decided to re-think the architecture of the system from the ground up. Under company IR&D funds, we developed an object-oriented framework for constructing radar

Figure 1 Sample Radsim Image for 1-meter resolution



simulations that we felt would allow us to build new radar simulations for other aircraft in a cost-effective manner. Ada95 (Ada 9x then) was just becoming usable with the GNAT compiler being developed and supported by NYU. This made it a natural choice to implement our object-oriented design.

The new architecture was based upon a melding of a custom object oriented version of the software structural model from the SEI and a Model-View-Controller architectural pattern. It was designed to consist of a proprietary core that contained the models that simulate radar effects, and open architecture view and controller modules that allow the radar to be customized to the look and feel of specific a radar and interfaced into existing host flight simulations.

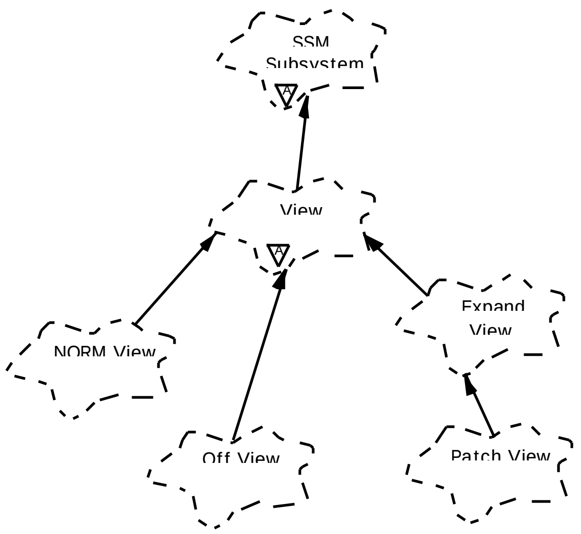


Figure 2 Class Hierarchy

An example of this is the basic view class hierarchy is depicted in figure 2. The base class is an abstract class we called a Software Structural Modeling Subsystem class that is the base class for all schedulable classes in our architecture. It allows us to add and subtract various schedulable objects to the DRLMS executive at run time via simple configuration text files. The basic View class is intended to encapsulate all the appearance features for the radar display produced by the DRLMS. The abstract base class View sets up a common interface for the view classes. There are a number of different types of radar formats used in aircraft radar. They can be pretty much categorized as Normal (or Real Beam) mode, Off (blank), and Expanded/Patch views. Different radars use one or more of these types of formats depending upon their operating mode. For example, the F-16 uses the Norm View to present the real beam mode, the Expand View to do the expanded real beam and range scale based DBS (Doppler Beam Sharpening) modes, and the Patch View do the patch size based DBS. These same classes work for a number of F-111 modes as well. The Patch View is also used for Synthetic Aperture Radar (SAR) modes like those found in a B-1 or B-2 radar simulation.

5. Current Work

This architecture has been used to develop a number of radars. Examples are the F-111 Attack Radar System (ARS), the F-111 Terrain Following Radar, the F-16 block 50, and a proof-of-concept demo that simulates a 1 meter per pixel resolution Synthetic Aperture Radar (SAR). The first of these, the F-111 ARS, was obviously the hardest as it was the test bed for the new architecture. The view and controller classes were developed for it from scratch as well. Subsequent radars proved to be much more straightforward, and the time to build new systems has been decreased substantially. Code reuse is strong as the core code is improved upon making it more flexible. The radar core remains both upward and downward compatible as its been improved, while much of the existing view objects have proven to be reusable as well as many radars share a common video format.

In addition, we have recently begun porting our DRLMS software from its original SGI host to other systems. This activity consisted of

- A few man days to bring the code up to where it was compatible with the current GNAT compiler version (from 3.04w to 3.10),
- Removal of a few SGI specific tasking directives,
- Approximately a week to redevelop the graphics output debug screens from SGI proprietary GL, to portable X,
- A recompile and a few more man days of debug and optimization.

This is not yet deliverable code. However, it was good enough for a successful customer demo, and we hope to complete the ports soon. The remaining work on the port is to optimize it for the new processors, and more fully test it. We expect it to amount to no more than a couple of man months to complete our ports and tests. It should be noted that as part of this port we are improving our portability, and the current port version code runs on SGI, Sun, Linux, and MacOS(MachTen) with no changes except for the debug graphics module.