Nu Game Engine (pre-pre-preview)

Bryan Edds, 2014

# Table of Contents

Table of Contents 2

What’s It All About? 3

Basic 3

Purely-Functional(ish) 3

2d Game Engine 3

F# 3

Getting Started 4

What is NuEdit? 4

# What’s It All About?

The Nu Game Engine is a **Basic**, **Purely-Functional**(ish), **2d Game Engine** written in **F#**.

Let me explain each of those terms –

Basic – Nu is very young, so it has just about no frills. Is there a particle or special effect system? Not yet, I’m afraid. Is there a sprite animation system? Again, not yet. However, there is a tile map system that utilizes Tiled#, and there is a physics system that utilizes Farseer Physics. Rendering, audio, and other IO systems are handled in a cross-platform way with SDL2 / SDL2#. In addition to that, there is an asset management system to make sure your game can run on memory-constrained devices such as the iPhone. On top of all that, there is a built-in game editor called NuEdit! So while there are plenty of missing features, you can see they might be worth building for yourself!

Purely-Functional(ish) – Nu is built on immutable types, and unlike with other game engines, data transformations and state transitions are implemented with copying rather than mutation.

Don’t mistake Nu for being slow, however. Notice I said Purely-Functional-ish. The ‘ish’ means that there are some imperative operations going on in Nu, almost entirely behind the scenes. For example, the Farseer physics system is written in an imperative style in C#, and some parts of Nu are optimized with imperative code as well. Fortunately, nearly all of this will be transparent to you as the user. When writing Nu code, feel absolutely safe, if not empowered, to write in the pure-functional style.

2d Game Engine – Nu is not a code library. It is a game software framework, and thus sets up a specific way of approaching and thinking about the design of 2d games. Of course, Nu is intended to be a broadly generic toolkit for 2d game development, but there are some design choices that will equally constrain and help you. Figure out how to leverage Nu’s design for your game. If it’s a complete mismatch, it might be time to consider using something else.

F# - So why F#? First, and foremost, its cross-platformedness. Theoretically, Nu should run fine on Mono for systems such as Android, iOS, OSX, \*nixes. It definitely runs on .NET for Windows. Note my weasel-word “theoretically”; Nu is still in such an early stage that it has yet to be configured, deployed, or tested on Mono. Nonetheless, since Nu only takes dependencies on cross-platform libraries, there should be no reason why it can’t with a little bit of appropriate nudging.

But more on why F#. F# is probably the best mainstream language available for writing a cross-platform functional game engine. Unlike Clojure, the static type system makes the code easier to reason about and dare I say more efficient. Unlike JVM languages, F# allows us to code and debug with Visual Studio. Finally, I speculate that game developers have more familiarity with the .NET ecosystem than the JVM, so that leverage is right there.

# Getting Started

Nu currently does not have a binary distribution. Instead it has a github repository at <https://github.com/bryanedds/OmniBlade>. I’m going to assume you know (or can quickly figure out) how to pull down the repository on your own. Please take note of the license when pulling down the repository.

The first thing you might notice about the repository is that it contains more than just the Nu Game Engine. It also includes the source for the Aml programming language, the Prime F# code library, and some other loosely related stuff. Both Aml and Prime are required to build the Nu solution, and the rest of the stuff is safely ignored, so feel free to pull it all down.

To open the Nu solution, first make sure to have Visual Studio 2012 (or maybe 2013 – not tested!) installed. Then navigate to the ./Nu/Nu folder and open the Nu.sln file. Attempt to build the whole solution. If there is a problem with building it, try to figure it out, and failing that, ask me questions via [bryanedds@gmail.com](mailto:bryanedds@gmail.com).

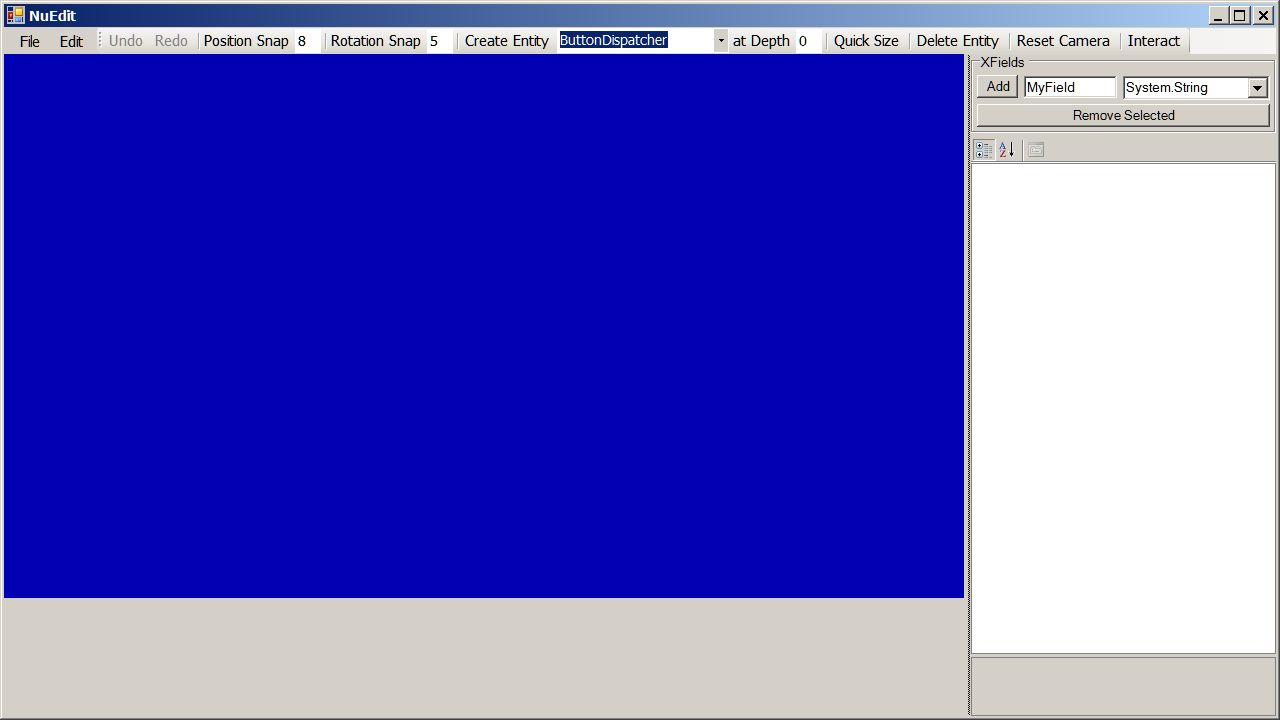
Once you have built the solution, try running the game I’m currently making by setting the Nu project as the startup project and running.

You

Now try running Nu’s game editor by setting the NuEdit project as the startup project and running. Playing with NuEdit will be a reasonable way to further orient you.

# What is NuEdit?

NuEdit is Nu’s fairly usable game editor. Here is a screenshot –



***NOTE:*** *There may still be some stability issues with NuEdit, so save your documents early and often!*