Mud Designer Documentation

Release 0.0.0.1

Johnathon Sullinger

May 21, 2017

Contents

Note: Don’t mind our dust, we’re still under construction. The documentation is a work in progress.

Contents:

CHAPTER 1

Getting Started

Note: Don't mind our dust, we're still under construction. The documentation is a work in progress.

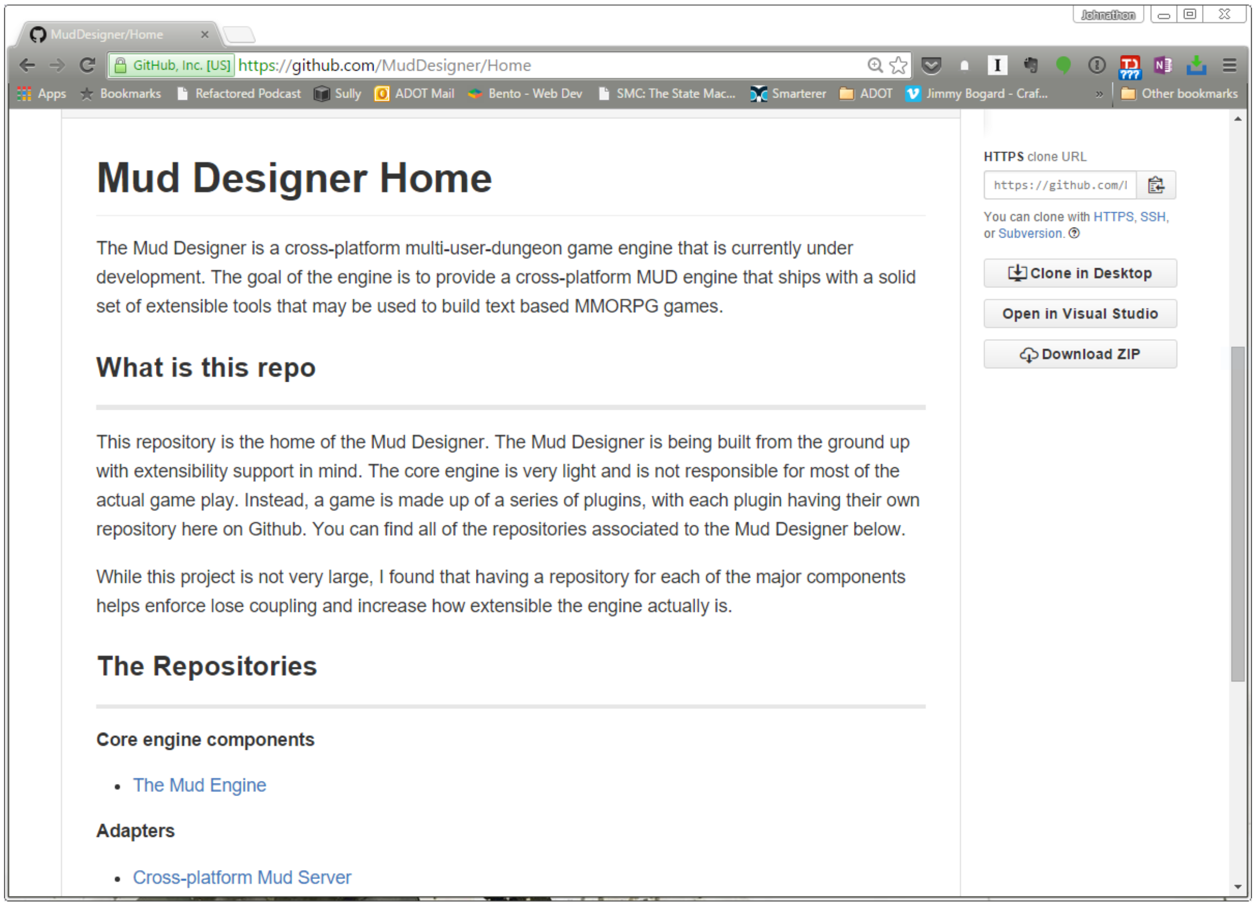
Setting up the Mud Engine on Windows

Note: Don't mind our dust, we're still under construction. The documentation is a work in progress.

As the engine is still under development and doesn't have an official binary release yet, you can clone the repository, compile it and use it to create your games.

Note: The engine is still under heavy development. There **will** be breaking changes.

You may visit the projects GetHub Home Page to see all of the available repositories you can clone.



Mud Designer Home

The Mud Designer is a cross-platform multi-user-dungeon game engine that is currently under development. The goal of the engine is to provide a cross-platform MUD engine that ships with a solid set of extensible tools that may be used to build text based MMORPG games.

What is this repo

This repository is the home of the Mud Designer. The Mud Designer is being built from the ground up with extensibility support in mind. The core engine is very light and is not responsible for most of the actual game play. Instead, a game is made up of a series of plugins, with each plugin having their own repository here on Github. You can find all of the repositories associated to the Mud Designer below.

While this project is not very large, I found that having a repository for each of the major components helps enforce lose coupling and increase how extensible the engine actually is.

The Repositories

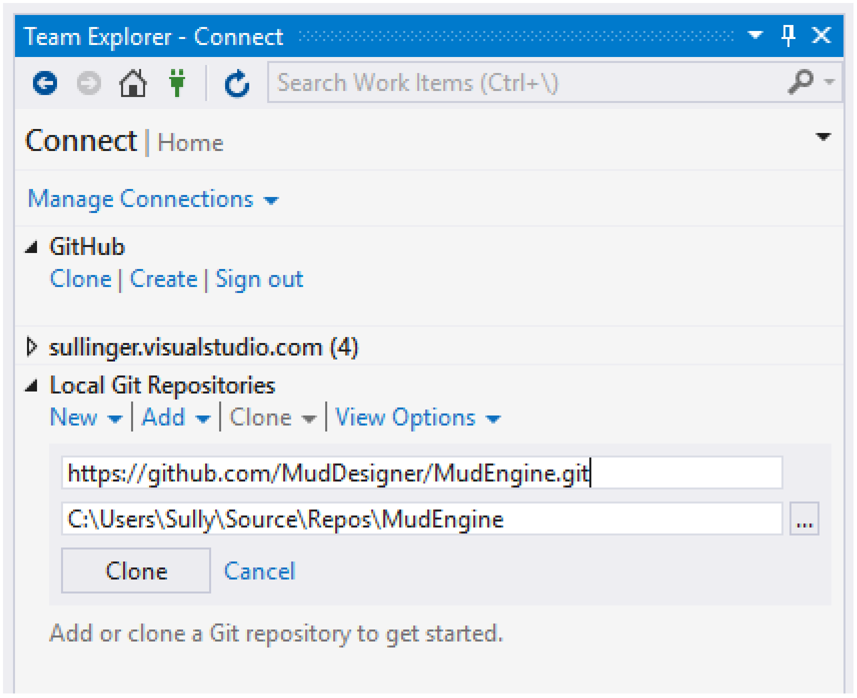
Core engine components

- The Mud Engine

Adapters

- Cross-platform Mud Server

To get up and running, you will need to clone the Mud Engine repository at a minimum, along with have Visual Studio 2015 Community Edition or greater installed. This will bring down the core engine components and a basic implementation of a game system. The Mud Engine repository does not include any servers.

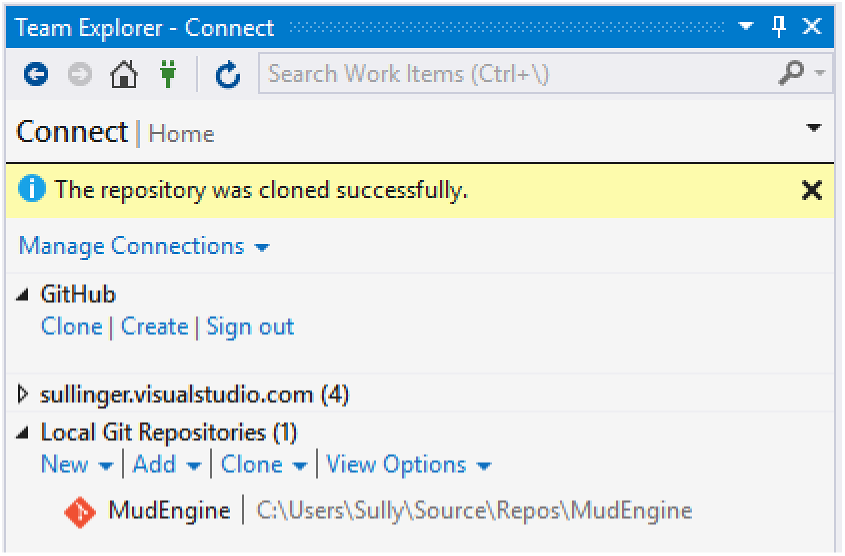

Team Explorer - Connect
Search Work Items (Ctrl+\)
Connect | Home
Manage Connections
GitHub
Clone | Create | Sign out
sullinger.visualstudio.com (4)
Local Git Repositories
New | Add | Clone | View Options
Clone
Cancel

https://github.com/MudDesigner/MudEngine.git

C:\Users\Sully\Source\Repos\MudEngine

Add or clone a Git repository to get started.

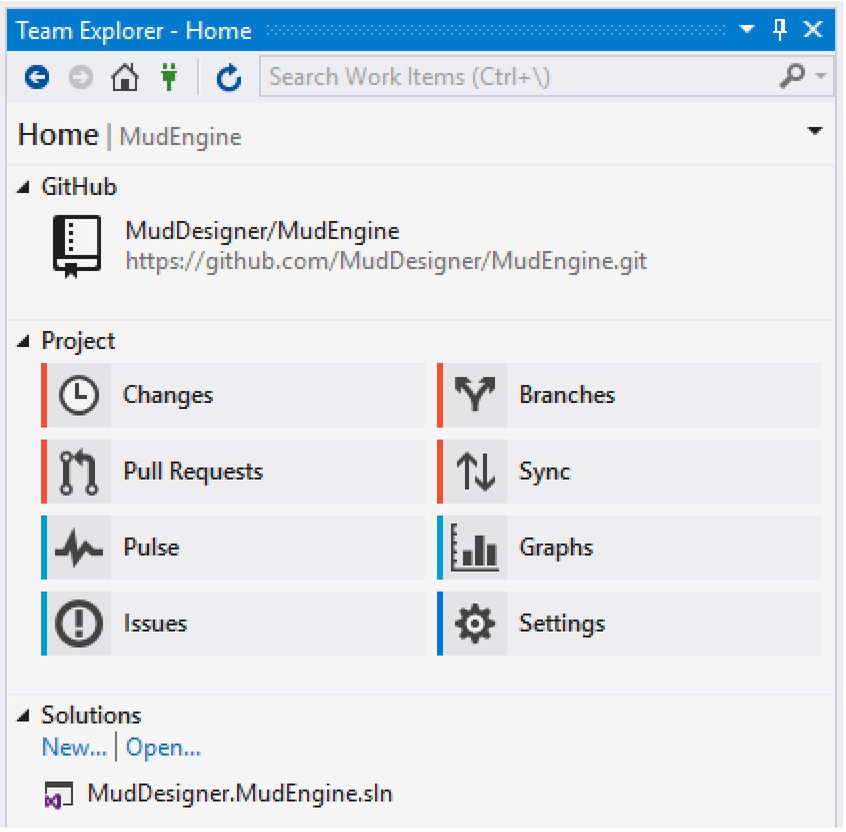
Once you have the repository cloned, you should see the repository available for you to view from within Visual Studio.


Team Explorer - Connect
Search Work Items (Ctrl+\)
Connect | Home

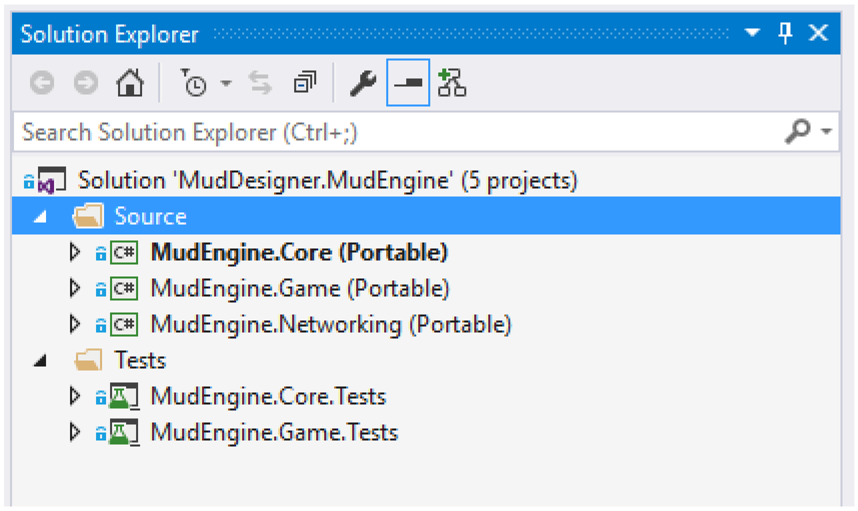
The repository was cloned successfully.

Manage Connections
GitHub
Clone | Create | Sign out
sullinger.visualstudio.com (4)
Local Git Repositories (1)
New | Add | Clone | View Options
MudEngine | C:\Users\Sully\Source\Repos\MudEngine

Opening the local Git repository will show you the Visual Studio solution file that you can open. If one does not exist, you can browse for one in the directory that the repository was cloned to. You will need to look for the *MudDesigner.MudEngine.sln* filename.



With the solution loaded, you should see all of the available projects associated with the Mud Engine solution.



Solution 'MudDesigner.MudEngine' (5 projects)
Source
MudEngine.Core (Portable)
MudEngine.Game (Portable)
MudEngine.Networking (Portable)
Tests
MudEngine.Core.Tests
MudEngine.Game.Tests

Rebuild the solution, and then navigate to the /bin directory within the *MudEngine.Core* project. You will need to copy the .dll files out of there when you create your new project.

Next, you will need to create a new project in Visual Studio. Once you have one created, add the compiled Mud Engine assemblies to your project as a reference. You are now ready to start building your own game.

Setting up the Mud Engine on OS X

Note: Don't mind our dust, we're still under construction. The documentation is a work in progress.

Understanding the architecture

The Mud Engine is being developed to be extremely flexible. The goal of the project was to allow any developer to swap out any component within the engine, without needing to open up the the engine source code.

In order to achieve this, there were a few design choices made that all Mud developers using the engine should be aware of.

Mud Engine Core

The core of the engine contains very little implementation logic. The idea behind the core is to provide a series of Types that would almost never be replaced. The rest of the content in the Core represents interfaces for other libraries to implement the details of.

An example of Types that would almost never be replaced would be the Mediator Types, some Exception and Argument Types along with some abstract base classes.

The core supports two different component types at runtime.

- Game components
- Adapters

A different project may implement the interfaces required by the engine as components, allowing different projects to have different sets of rules and behaviors. Components make up things that run in parallel with other objects in the game, but typically always on the engines main thread. An example of this would be characters, equipment and rooms.

A project may also adopt the Adapter system of the engine. Adapters can run on their own background thread if they want. Adapters allow a project to add complete new systems to the game, that were not originally accounted for by the engine. An example of this would be the world time, a networked server and complex persistance systems.

Mud Engine Game