Verigames User Manual

Welcome to Verigames! We are a group of students at the Paul G. Allen School of Computer Science and Engineering who are interested in bringing code verification to everyone! In this document you will find the resources you need to start verifying programs yourself.

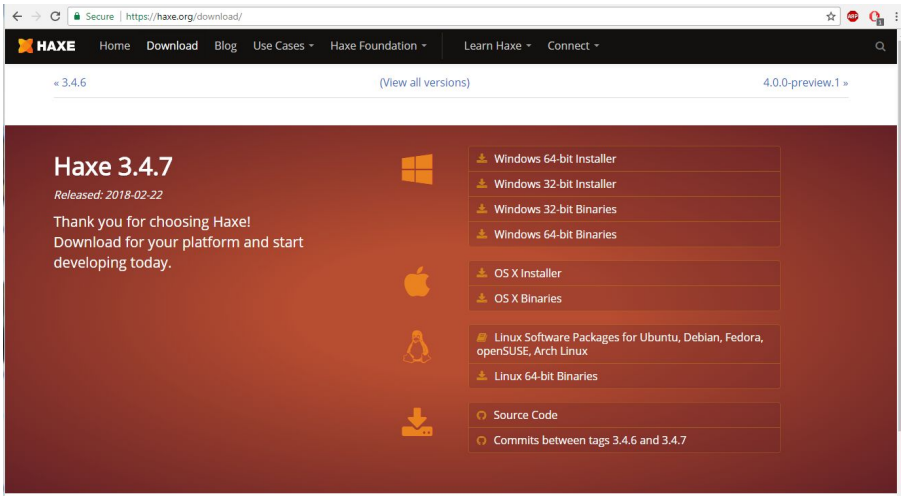
Let's start with what verification of a program means. If you are not familiar with the term, formal verification is a way of proving that a program meets a certain specification (a set of requirements). This process is usually very time consuming because it requires a trained engineer to go through possibly thousands of lines of code, line by line, verifying that the specification holds at all points in a program. For example, if you had a program and wanted to ensure that it did not try to divide by a number by zero anywhere (a common way to cause a program to crash), one would need to go through a whole program and prove with formal mathematics that that cannot occur.

What Verigames does is take all of this formal mathematical logic and wraps it up in a game that is fun and challenging so that ANYONE (even those without any knowledge of programming or software engineering!) can help verify programming. In this way, you and anyone else can help prevent costly software errors in systems where a crash can mean a significant loss of resources or even a loss of human life, such as software in an airplane or a hospital.

If you are someone who just wants to play and help make the world a safer place, you can stop reading right here! Our verification game (PipeJam/FlowJam) can be found on both the Apple App Store and the Google Play Store. A short tutorial in the game will introduce you to the game and teach you how to play. If you are a developer who wants to build on the project or submit code for verification, read on.

Verigames takes a Java program and a type annotation system to represent the specification you want to use (For more information on this see `verigames/Java/USING_GRADLE` and `verigames/documents` in the Verigames GitHub repository).

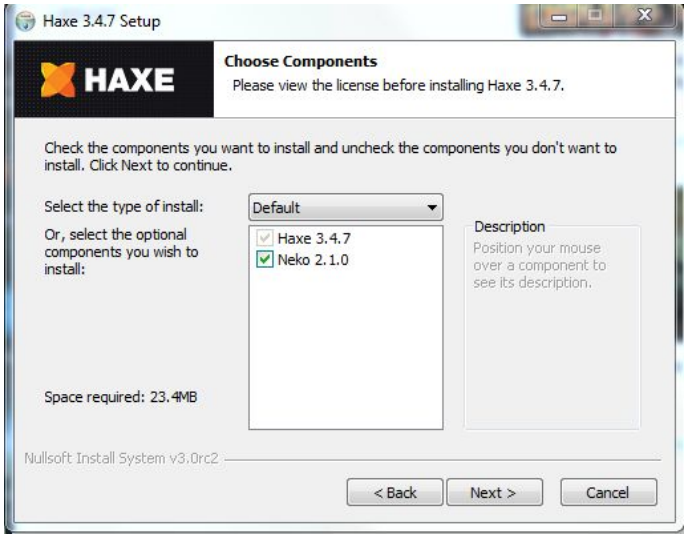
Downloading Haxe/Develop and necessary libraries and compiling



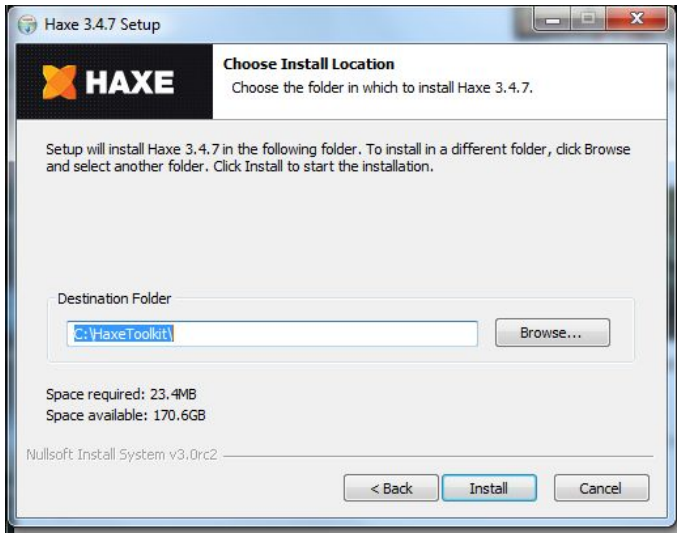
1. Download Haxe for your system. We recommend using the installer and not the binaries for ease of installation.

Next install haxe by running the executable you downloaded.

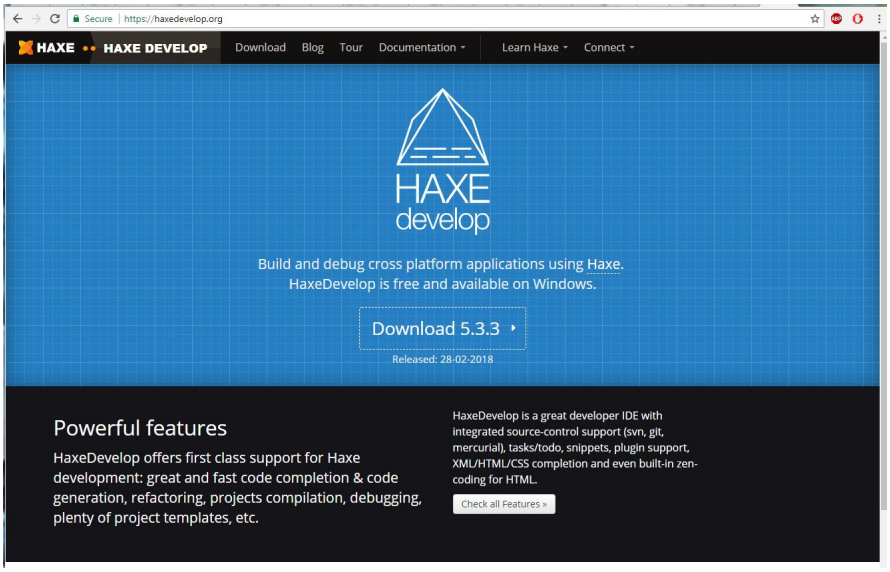
This window will appear, click “Next” to continue installation.



The next window will have you select installation type, we recommend using default and installing Neko if you do not already have it on your system. Click the “Next” button.



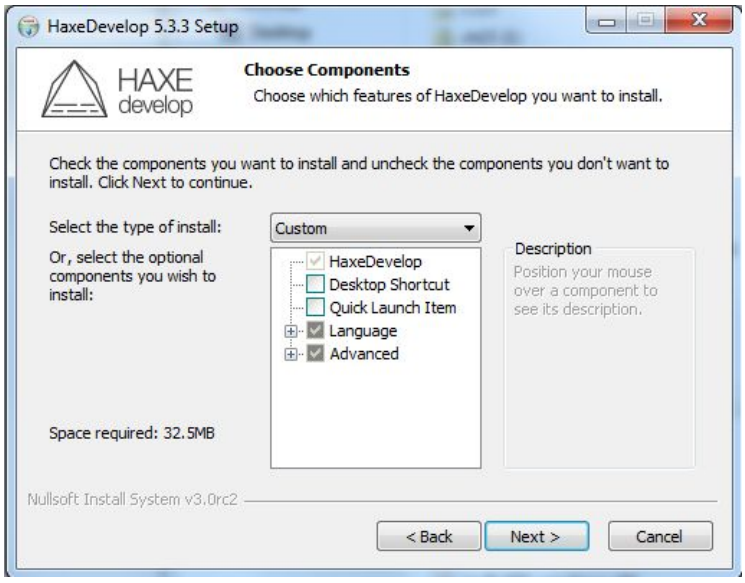
Next choose the installation destination and click “Install.”



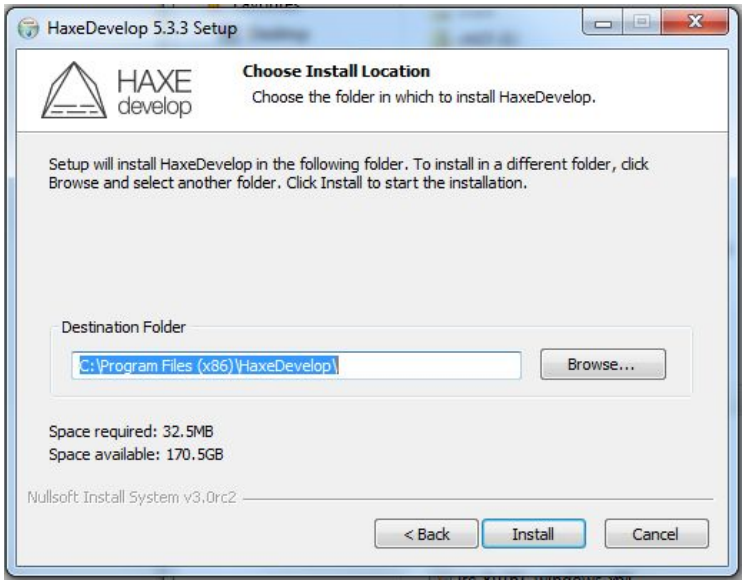
2. Navigate to www.haxedevelop.org and click the download button.

Note: HaxeDevelop is Windows only. For setting up a Haxe IDE on other OSs, see [here](#).

Navigate to the file you just downloaded and double click on it. Click “Next” once this window appears.



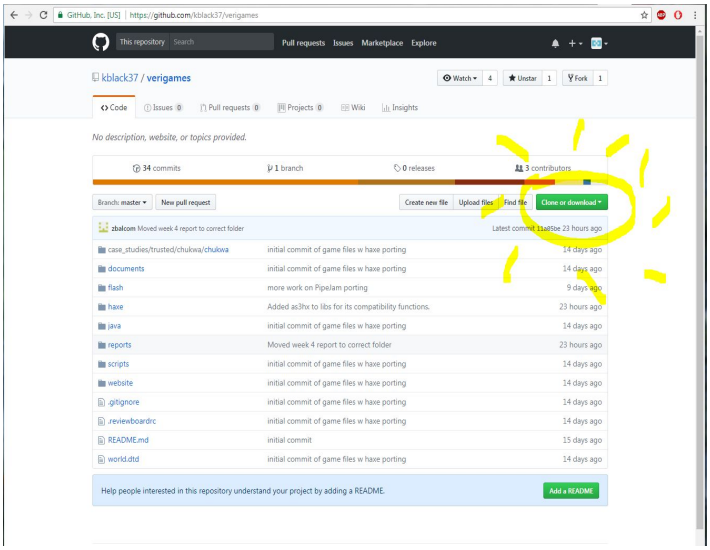
Click “Next” again after this window appears. Leave Language and Advanced boxes ticked.



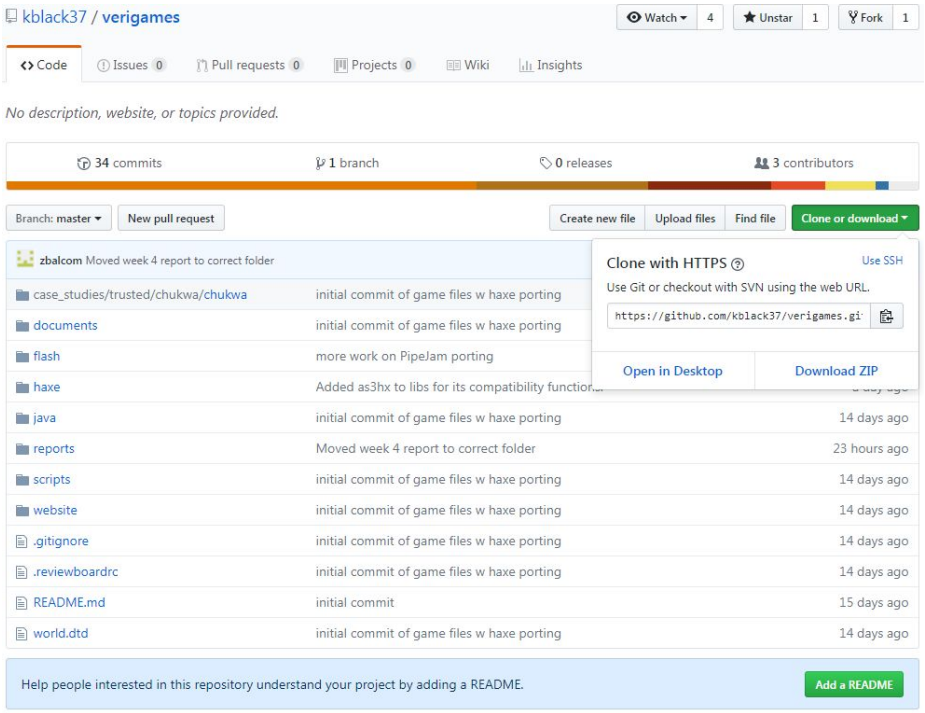
Choose your installation location and click “Install”

Once these have been installed you still need four more resources to use Verigames. Open your command line interface and execute the following commands:

```
haxelib install lime
haxelib install nme
haxelib install openfl
haxelib install starling
haxelib install as3hx
```



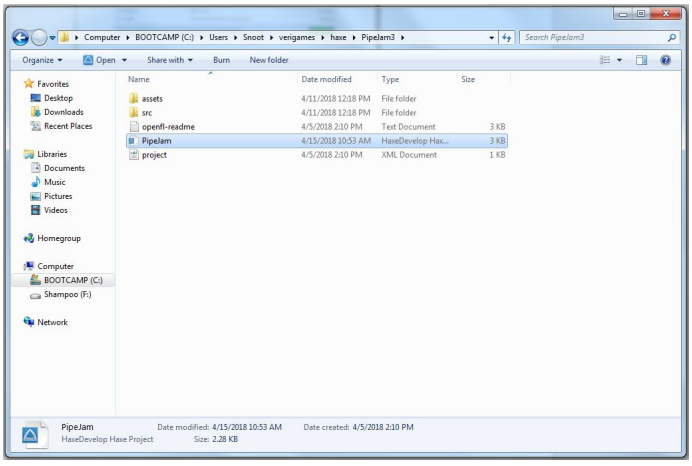
3. Assuming you are viewing this from our GitHub, next you will need to clone the repository. If not then visit us at https://github.com/kblack37/verigames.



Click the clone or download button and choose the option you want to get our repository.

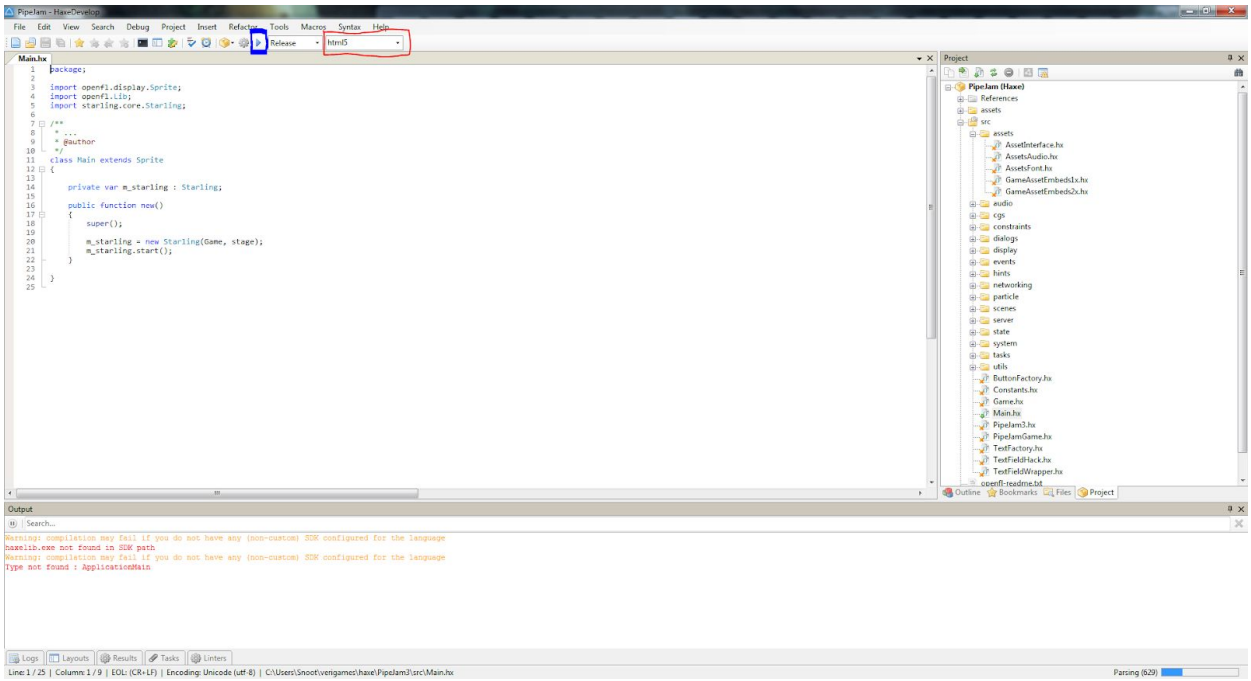
Alternately, you can use the command line if you have git installed to clone the repository using the command:

```
git clone https://github.com/kblack37/verigames.git
```



4. Now that you have the Verigames repository cloned to your machine, you can navigate to `verigames/haxe/PipeJam3` and open up `PipeJam.hxproj`. This will launch HaxeDevelop with the PipeJam project open.

5. From the first window navigate to `src/Main.hx` and double click that file. Then hit the play button (shown in blue) to launch the game in HTML5 or choose a different platform to build it in (shown in red).



Submitting Your Own Code for Players to Verify

1. Ensure Python 2.7 or later is installed and is added to your PATH.

2. Follow the instructions for running the Java tool to generate JSON files that are then translated into levels. These are located in the documents folder in the Verigames repository.

3. To send the generated JSON files to the Verigames server for translation into a game world and insertion as a playable level, run:

   ```
   python worldsubmit.py -u [username] -p [password] -f [path/to/level/json]
   ```

   If you do not have a current username or password, enter a new one at this time. If your username is already in use, you will be prompted to choose a new one.

4. If you wish to see the results of the verification so far, run:

   ```
   python worldresults.py -u [username] -p [password] [-d dir]
   ```

   The results will be output in the target directory or in the default directory out in the Verigames repository.