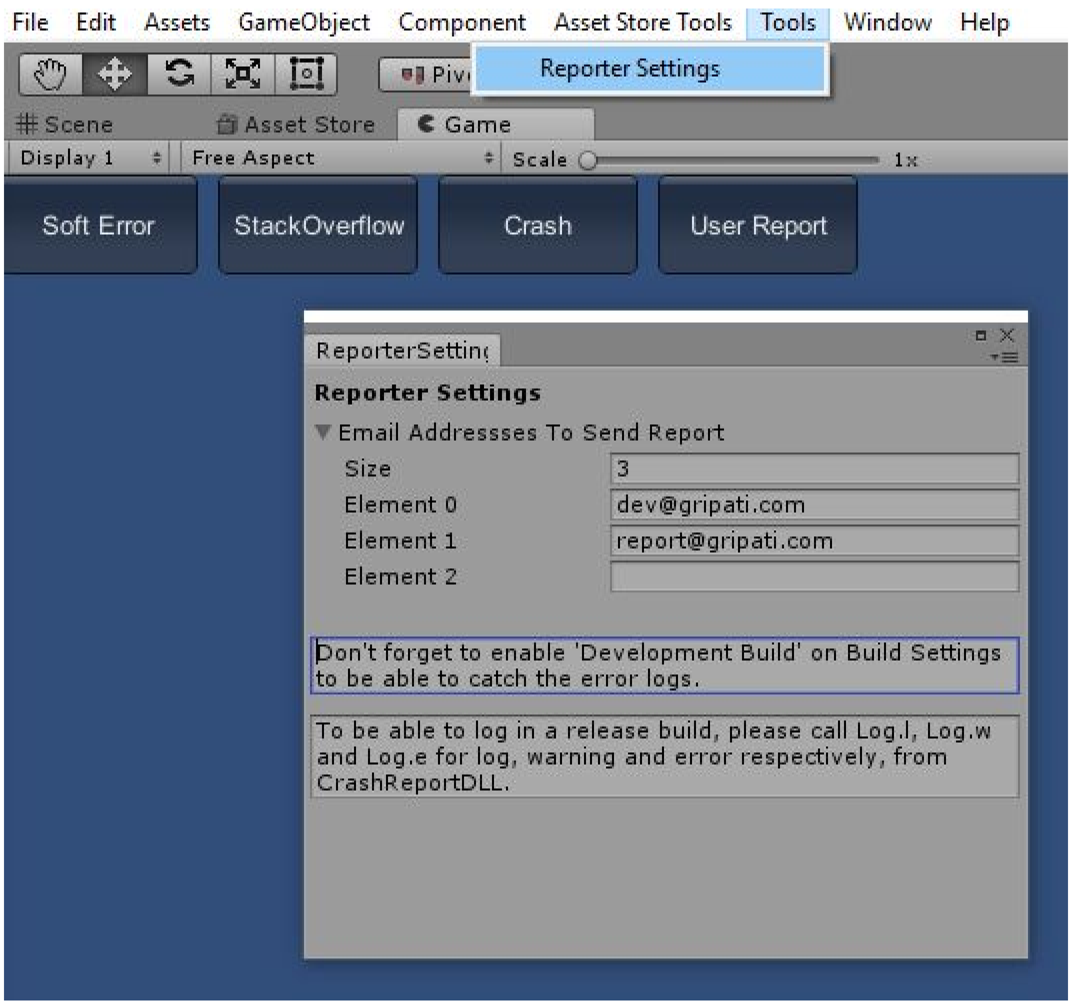
Reporter

Unity integration and Backend service for user feedback and crash report with screenshot.

There is no need to do any coding to activate Reporter in your project. After adding the Reporter plugin to your project you should set up Reporter on Tools>Reporter Settings. This will open a popup to set the email addresses.



Reporter Settings

Email Addressses To Send Report

Size	3
Element 0	dev@gripati.com
Element 1	report@gripati.com
Element 2	

Don't forget to enable 'Development Build' on Build Settings to be able to catch the error logs.

To be able to log in a release build, please call Log.l, Log.w and Log.e for log, warning and error respectively, from CrashReportDLL.

"**Email Addresses to Send Report**" is an email list. In this version you can set max 3 email addresses to send the report. The email addresses should be valid, or the setting window would warn you.

Reporter will store the screenshot images on our server to serve you as a link at the email you receive. These images are stored in the way linked to your unity username, so that to make it sure only you can reach this images. Furthermore, the names of the images on our servers is hashed to avoid availability in case of a brute force attack.

Reporter adds Report.cs to “Script Execution Order”. This assures to catch the errors and the crashes which are caused in the application load.

Testing

A testing script is included. Simply attach the script called “Test.cs” to any GameObject on the scene. Afterwards when you run your project, you should see three buttons on the top-leftmost of the screen:

1. **Soft Error:** Creates a NullReferenceException.
   Reporter will take a screenshot and send the logs when an exception is fired. This is done in an separate Thread, not to harm the app performance.

2. **StackOverFlow:** Create StackOverflow Exception.
   Mostly, Reporter will be able to send a report in this case. But on a device a StackOverflow might crash the app. If it crashes, Reporter will track the report sending progress and will send it next session if it fails in this session. You might receive two reports in this situation, because the app might die before the response of the report request is received. A crash report will still be sent next session to be sure not to miss any reports.

3. **Crash:** Crash the app immediately.
   This will force unity to create an inner crash and is only working inside Unity Editor. Even though this isn't a real life scenario, Reporter will recognise the crash of the app and report the log next session. In this case a screenshot won't be possible.

4. **User Report:** Send a user report.
   First capture a screenshot so that the scene reflects the state of the user complain without the report window. After the screen pixels are read, open the user report form (this is a prefab included to the project) and ask the user for a feedback.

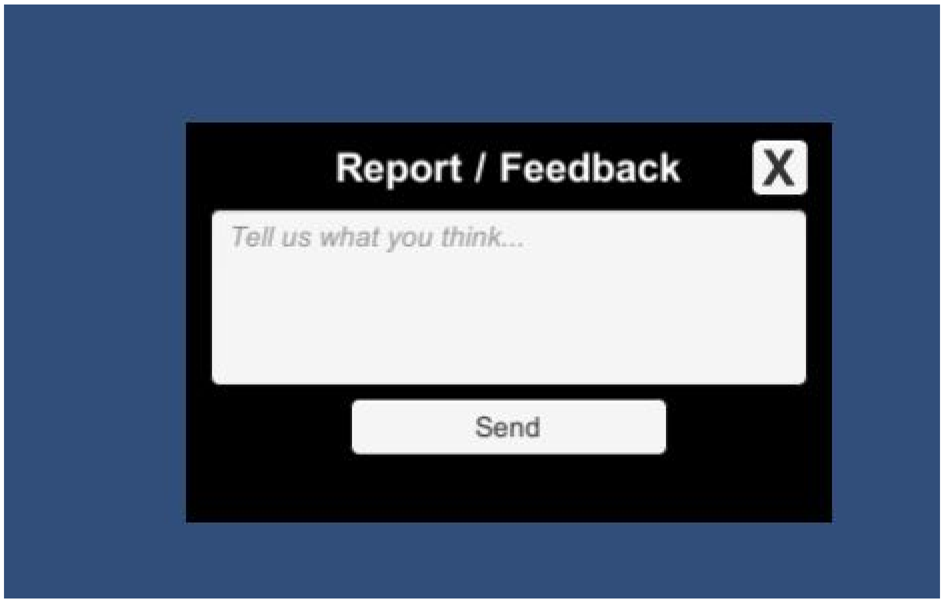
Usage

Reporter can be used for two missions:

1. **Crash Report:**

Reporter starts to listen to your application on startup and logs the Debug traces of your application. After setting at least one email address in the “Report Settings”, Reporter will automatically catch the exception in the published game, or test environment, or unity simulator. After catching the exception, Reporter will save the Debug logs and the stack trace of the exception. Also Reporter will take a screenshot and send all these information as an email to the listed email addresses.
You will find a link to the screenshot at the moment of the exception and all the Debug logs (Debug.Log, Debug.LogWarning and exception stack trace) with time and type in well formatted style.

2. **User Feedback:**



You can also use Reporter as a service without a crash. You can use the simple function in Reporter.cs to send a user feedback to the listed email addresses:

```
public static void SendUserReport(string title="", string message="", string userId="",
                                        Action<bool> onComplete = null)
```

You can call this function with optional 4 parameters: Title, message, userId and onComplete. Depending on the way you ask a feedback from the user you can feed any of these three parameters to this function. After the function call, Reporter will take a screenshot and send an email with the given parameters and the Debug logs.

Be aware! Probably you should call *public static void CaptureScreenShot(Action onScreenShotCaptured = null)* **before** calling *SendUserReport*. If you are going to show a window to the user to fill some message for the feedback, you should first take a screenshot by calling *Reporter.CaptureScreenShot()*. This way you will get a clear image of the application at the moment the user decides to report. Otherwise your feedback form might be over the data you are looking for on the screenshot.
Sample usage:

```
void OnUserClickedFeedbackButton()
{
        Reporter.CaptureScreenShot(onScreenShotCaptured);
}

private void onScreenShotCaptured()
{
      GameObject userReport = Resources.Load<GameObject>("UserReport");
      userReport = Instantiate<GameObject>(userReport);
}
```