DubitaC reference manual

`https://github.com/WeLikeIke/DubitaC/tree/v1.3.0`

Lorenzo Tibaldi

March 2022

Contents

1 Reference manual **2**
1.1 How to play . . . 2
1.1.1 Players & Clients . . . 2
1.1.2 Admin & Server . . . 3
1.2 How to maintain . . . 4
1.2.1 Small tweaks . . . 4
1.2.2 Extending existing systems . . . 4
1.3 Troubleshooting . . . 6
1.3.1 Players & Clients . . . 7
1.3.2 Admin & Server . . . 9
1.3.3 Shared . . . 10

Chapter 1

Reference manual

The reference manual offers some guidance for how to play and how to maintain DubitaC, with an additional troubleshooting section at the end.

1.1 How to play

Important parts are written in **bold**.

1.1.1 Players & Clients

Setup phase

- Setup the preferred localization and options.
- Insert the server Ipv4 address, **must be shared by the server admin**.
- Insert the user credentials, if you do not have an account yet, insert the user credentials as normal but check the checkbox "Create new account".
- Press the "Play" button and avait for the connection to be accepted.
- Select an avatar between the available ones.

In case the connection is refused, for any reason, it will be possible to retry as much as needed.
The solution creation round will be loaded when the server admin decides to start it.

Solution creation round

- Read the description of the codeQuestion.
- Type the solution in the notepad in the centre of the screen, **do not change the function signature that is already written**.
- Test your solution by pressing the button "Test".
- Check the test results by pressing the button "Log".
- If needed, improve your solution to pass more tests.
- If available, press the button "Ready" to stop writing the solution and save your position in the leaderboard.
- At any moment, by pressing the "Hints" button, a panel containing the description of the codeQuestion and the possibility of buying hints will be opened.
- At any moment, by pressing the "Export" button, a copy of the solution will be saved on the local machine.

The doubting round will be loaded after the available time expires.

Second round

- Select one of the user avatars on the left to be able to read their solutions.
- If you spot a mistake, press the "Doubt" button to open the doubt panel.
- Create a doubt by filling out the panel and confirming with the "Doubt" button, **only one doubt can be created towards each user**.
- To remove a previously created doubt, select the target user and, in the doubt panel, press the top right button to remove any doubt towards the selected user.

A loading screen will appear when the available time expires.
The slideshow will be loaded after all clients and server have completed their executions.

Slideshow

- Observe all doubts that will be shown during the slideshow.

The intermediate leaderboard will appear when the slideshow finishes.
The final menu will be loaded when the server has completed the final executions.

Final evaluation phase

- If needed, by pressing one of the 2 "Export" buttons, the own solution or the best solution of the lobby will be saved on the local machine.
- Disconnect by pressing the "Disconnect" button in the top left corner.
- Go back to menu by pressing the "Back to menu" button in the top left corner.

The final evaluation phase shows the final leaderboard of the game session.
After having disconnected, a panel showing the amount of current points will be shown.
The setup phase will be loaded after pressing on the "Back to menu" button.

Tips

The cost of the hints is minimal, always make sure to use them when some help is needed.

The list of users on the left during the doubting round is ordered using the current leaderboard, it is more advantageous to start doubting from the top since that will increase the chances of bringing them down.

While examining a solution, if no mistake can be found, it is more efficient to try to analyse another solution instead of trying to create a doubt for the current one.

1.1.2 Admin & Server

Setup pahes

- Setup the preferred localization.
- Share the Ipv4 address that is shown at the top of the screen, **this step is required**.
- If needed, insert the amount of available time to solve the codeQuestion.
- Select a codeQuestion from the list on the right, the tag filter can be used to quickly narrow down the number of options, **always select a codeQuestion that can be solved by the players based on the knowledge that they should have**.
- Start the game session, **At least 2 clients must be connected**.

The solution creation round will be loaded when the "Start" button in the bottom right corner is pressed.

Solution creation round

- While waiting for the available time to run out, a panel showing the current progress can be read.

The doubting round will be loaded after the available time expires.

Doubting round

- While waiting for the available time to run out, a panel showing the current progress can be read.

The slideshow will be loaded after all clients and server have completed their executions.

Slideshow

- While waiting for the available time to run out, a panel showing the current progress can be read.

The final evaluation phase will be loaded when the server has completed the final executions.

Final Evaluation

- If needed, by pressing the "Export" button, all user solutions will be saved on the local machine.
- Shutdown the server by pressing the "Disconnect" button in the top left corner.

The setup phase will be loaded after pressing on the "Disconnect" button.

Tips

The admin interaction is minimal after the setup phase, however, after the final evaluation phase has been loaded, it is important to save the user solutions on the local machine, so that they might be analysed later. For this purpose, the user solutions will contain a first comment line with the username of their author.

It might be wise to write somewhere the usernames that are involved in a game session, alongside their real names, for easier feedback distribution.

1.2 How to maintain

Except for troubleshooting, see Section 1.3, the maintenance of DubitaC should involve small tweaks to the source code to best fit the current needs, or extension of already existing systems.

DubitaC was developed using Unity version 2020.3.30f11[1], which is considered the 2020 LTS (Long Term Support) release, meaning that newer features and fixes will be available until March 2023. Afterwards, it is recommended to migrate the game to a newer Unity version.

1.2.1 Small tweaks

Every script that contains a class contains at the beginning of its property declaration some constants or readonly values that can be modified to apply small changes in the game executions.

All classes that contain such constants or readonly variables will explain their usage in the class summary at the top of the file, see Figure 1.1.

1.2.2 Extending existing systems

Extending the localization

The localization system is completely managed by the Unity localization package.[2]

To introduce a **new** localization you would need to provide:

- A sprite of the country flag representing the language being added.

[1] Unity editor available here: `https://unity3d.com/unity/qa/lts-releases`

[2] Package manual available here: `https://docs.unity3d.com/Packages/com.unity.localization@1.0/manual/index.html`

```
/// <summary>
/// Class that manages the notepads, implements autocentering and Undo-Redo.
/// The maximum amount of zoom possible is stored in constant <see cref="maxAllowedZoom"/>.
/// The minimum amount of zoom possible is stored in constant <see cref="minAllowedZoom"/>.
/// The zoom granularity is stored in constant <see cref="zoomStepSize"/>.
/// The offset required to avoid that the cursor goes off screen while typing is stored in constant <see cref="typingOffset"/>.
/// </summary>
public class NotepadManager : MonoBehaviour {
    private const float maxAllowedZoom = 2f;
    private const float minAllowedZoom = 0.2f;
    private const float zoomStepSize   = 0.2f;
    private const float typingOffset   = 150f;
```

Figure 1.1: Example of a class that explains the available constants in the summary at the top.

- A localization CSV file containing "label - comma - localized text" in each row.

From then on, it is only necessary to add the support for your chosen locale and to update the relative tables:

- The "CountryFlags" table by adding the new sprite.
- The "Strings" table by importing the CSV.

To extend **existing** localizations, simply insert the new labels in all localization files following the localization format and then reimport all CSV files in the "Strings" table.

Extending the persistency system

The persistency system is completely managed by the "AccountManager" class, any modification to this class, that does not extend to other classes, must respect 2 restrictions:

- A function, that is public, void and at maximum with 1 parameter, needs to be exposed so that the submission of user credentials can be executed by a button being pressed by the player, currently this function is called "Submit".
- A function with the same signature as the "ValidateLogin", see Figure 1.2, must exist; so that the connection with the server can be validated using it.

If the first is not respected, the players will not be able to connect easily by pressing a button on the interface.
If the second is not respected, the server will accept any client that tries to connect, removing the user authentication step.

```
/// <summary>
/// Function required to validate the connection to the server.
/// The connection will be accepted if:
/// The sent username and password are valid OR
/// The sent username is new and the newAccount boolean is true.
/// And refused otherwise, if someone with the same username is already connected to the server then the newest client is rejected.
/// </summary>
/// <param name="connectionData">Array of bytes representing the data sent from client to server to validate the connection.</param>
/// <param name="clientId">Id of the client that requests a connection with the server.</param>
/// <param name="callback">Mandatory callback parameter, it signals to the server the outcome of the validation.</param>
private void ValidateLogin(byte[] connectionData, ulong clientId, NetworkManager.ConnectionApprovedDelegate callback) {
```

Figure 1.2: Signature of the function used to validate a client login, shown alongside its summary.

Extending the cosmetics system

The cosmetics system is managed by the static "Cosmetics" class for sprite initialization, while other classes are free to query it using its getters, for example, classes "AvatarUI", "AvatarManager" and "NetworkWrapper".
To introduce a **new** avatar:

- Create a new sprite with:
  - A name following the format: name of the sprite, underscore, number of points required to unlock.

  - Size of 64x16 pixels, divided into 4 squares of size 16x16, each containing a different avatar expression: normal, obscured, angry, happy.

- Insert the sprite in the folder at path: Assets/Resources/avatarSprites
- Modify the value of the image in the Unity inspector to convert it from a texture to a sprite, see Figure 1.3.
- Press the "Sprite Editor" button on the inspector to slice the complete sprite into the 4 smaller sprites.
- Open the txt file at path: Assets/Resources/avatarNames.txt
- Add the name of the new avatar **in a new line**, making sure that the complete name does not contain the extension.

To change the name or point threshold of **existing** avatars:

- Open the txt file at path: Assets/Resources/avatarNames.txt
- Change the name of the target avatars, making sure to respect the same format.
- Rename the avatar sprite in path: Assets/Resources/avatarSprites

Make sure that the avatar sprite name and the name inserted in the "avatarNames" text file always match.

Extending the codeQuestion system

The codeQuestion system is managed by the "CodeQuestionManager" class for the codeQuestions initialization. The "CodeQuestionUI" class is responsible for holding and displaying a single codeQuestion during the codeQuestion selection. Later, the "ExecutionManager" class , during startup parses the selected codeQuestion following the format detailed in Section **??**.

To insert a **new** codeQuestion:

- Create and test the cpp file that corresponds to the reference solution to the codeQuestion.
- Follow the conversion steps detailed in Section **??**.
- Insert the file in path: Assets/Resources/CodeQuestions

If the codeQuestion contents are malformed, an error is returned and continuing the execution of the game session might not be possible.

Make sure that the codeQuestion returns the expected values for all the tests.

To change the codeQuestion parsing, for example, to add a possible label, the "ExecutionManager" class would need to be modified.

Extending the main execution

The main execution is completely managed by the "ExecutionManager" class, thanks to the procedure of spawning a parallel hidden process to compile and test user solutions.

This is achieved thanks to the .NET namespaces **System.Diagnostics** that let us create new processes and **System.Threading.Tasks** that let us wait asynchronously for a result, see Figure 1.4.

From here, modifications regarding which processes to spawn can yield any desired result, for example, extending the commands to work on other platforms, see Figure 1.5.

Make sure that modifications to the process spawning process are secure and contained, since this part steps out of the Unity controlled environment, to avoid system problems.

1.3 Troubleshooting

Actions that can be taken to tackle each problem are written in **bold**, while additional information is displayed in normal text.

1.3.1 Players & Clients

There is no place to insert my user credentials

Only a client build will display the login menu.

Make sure that the build is a client build and not a server build.

When DubitaC is run, the first screen will contain a button with either "Continue as Server" or "Continue as Player" written on it, to be able to play the latter message should be displayed.

I cannot create a new account

It is possible to create a new account only if the credentials requirements are met:

- The username should be between 1 and 20 characters long.
- The username cannot be already in the database.
- The username cannot contain commas or whitespace.
- The password must be between 8 and 20 characters long.

Make sure that all requirements are met when inserting new user credentials.

I forgot my user credentials

DubitaC does not store passwords in a retrievable way.

In case of forgot password:

- **Ask the server admin to manually delete the relevant entry in the database and save the number of points that were earned.**
- **Recreate the account by making sure to check the "Create new account" checkbox and inserting the new username and password.**
- **Ask the server admin to manually modify the new entry in the database by inserting the old "points" value in the correct column.**

In case of forgot username: nothing can be done to retrieve the points that were earned, **create a new account**.

After pressing "Play" the connection takes a long time and fails

To be able to press "Play" a player has to insert a valid Ipv4 address in the relevant field, however, there is no guarantee that a server will be ready to listen on the other size.

Make sure that the inserted Ipv4 address is the same as the one that the server admin has shared.

Additionally, there might be connection problems on the local machine.

Make sure that firewall and antivirus do not interfere with the outgoing connections.

I cannot select some of the avatars

The avatar selection is tied to the progression with the game, some avatars will be obscured and non-interactive until a certain threshold of required points is reached.

Play more to unlock more avatars.

I could not choose my avatar in time

A random avatar will be assigned.

If this happens frequently, **ask the server admin to wait a bit longer before starting a game session**.

I want to play with a friend but we are never in the same lobby

The lobby rebalancing system might remove some clients from a lobby to place them in another.

Clients that connect later are moved more often, **try to connect as soon as possible**.

I cannot find my ".cpp" file after I pressed on "Export"

The creation of permanent ".cpp" files will depend on the path that was inserted in the options menu.

If none were inserted, a default path has been used.

Check the directory at path: %AppData%/LocalLow/LorenzoTibaldi/DubitaC

The log is not in my chosen language

Because of some implementation limitations, the log cannot be localized in the current version.

What is written in the log are either easy sentences or what the terminal returned on the local machine.

The game hanged

To check if the game is hanging, look for moving parts in the interface, for example, the loading screen or interactive intermediate leaderboard.

If nothing moves as expected, then the game hanged, **after waiting an appropriate amount of time, close the application since the game will not be able to be resumed**.

If the moving parts work, then the game is simply compiling and executing, **wait until the scene changes**.

I think my solution should not have timed out

The game will abort any execution that takes more than TIMEOUT amount of seconds, where TIMEOUT is the value that was selected in the options menu.

If none were selected, a default value of 3 seconds has been used.

In case a valid solution requires more than 3 seconds for an execution, **select a higher TIMEOUT value before starting the next game session**.

The maximum selectable TIMEOUT value corresponds to 9 seconds but if the solution takes longer we suggest **finding the parts of very inefficient code and refactor them**.

During the slideshow, a doubt reported that a solution returned "Unknown value"

Because of the current dependency on Catch2, there is no way to detect what a function returned when a test expected a crash or timeout.

In the future, it could be possible that a new Catch2 version would offer tests that, even when checking for exceptions, could capture the returned value of a function, at that point, it would be possible to extend the result parsing so that "Unknown value" will not have to be used anymore.

I do not understand the doubt evaluations

The doubts can fall into 4 categories:

- Correct doubt, meaning that the doubter correctly guessed the output of the reference solution and the output of the "wrong" target solution when given a certain input.
- Wrong doubt, meaning that the doubter incorrectly guessed the output of the reference solution and the output of the "wrong" target solution when given a certain input, while the target solution returned the correct result.
- Both wrong, meaning that the doubter incorrectly guessed the output of the reference solution and the output of the "wrong" target solution when given a certain input, while the target solution returned a wrong result.
- Half doubt, meaning that the doubter correctly guessed only one of the outputs between the reference solution and the "wrong" target solution when given a certain input, while the target solution returned the correct result.

The reasoning behind the evaluation is that a doubter to be awarded points needs to be able to guess the correct execution of the codeQuestion and the wrong execution of another player, while a solution that is being doubted will only need to return the correct result and it will be awarded some points.

My final position in the leaderboard is wrong

The leaderboard cannot be wrong, remember that there are 3 different orderings of the leaderboard:

- The order of players based on their order of pressing the ready button in the first round.
- The order of players based on the points that they received after testing all the "[user]" doubts.
- The order of players based on the points that they received after testing all the "[final]" tests.

Between the end of the slideshow and the final leaderboard, the order can still change, depending on the final tests that are executed.

How many points did I win?

The number of points that players obtain after each game session depends on 2 factors:

- The number of hints bought during the first round.
- The final position on the leaderboard.

There is always a minimum number of points that will be awarded for each game session, but the higher your position in the final leaderboard, the more points will be awarded.

Additionally, in the final menu after pressing disconnect, it is possible to see how many points the user currently has without having to start a new game session.

When is it safe to disconnect?

The game saves user progress on the database after having created the final leaderboard, this means that **it is safe to disconnect once the final menu has been loaded**.

A disconnection during the slideshow will *not* result in your progress being saved.

I was kicked back to the main menu

Clients are sent back to the main menu when they lose connection with the server.

All clientside and serverside disconnections are handled gracefully, but in the case of a clientside disconnection *the game might not continue for the clients that are still connected*, see Section 1.3.3.

1.3.2 Admin & Server

There is no place to select the codeQuestions

Only a server build will display the codeQuestion selection menu.

Make sure that the build is a server build and not a client build.

When DubitaC is run, the first screen will contain a button with either "Continue as Server" or "Continue as Player" written on it, to be able to select the codeQuestion the former message should be displayed.

Can I move the clients between lobbies manually?

No, manual lobby rebalancing is not supported in the current version.

I forgot to set the available time

A default amount of 600 seconds will be set as the available time.

In case 10 minutes are too long or too short, **close and reopen the application, so that a new game session can be created**.

I cannot find user solutions after I pressed on "Export"

The user solutions are saved in the default path.

Check the directory at path: $InstallationDirectory$/DubitaC_Data/Cpps

Where $InstallationDirectory$ is the folder where the game is installed, the one containing the file DubitaC.exe.

I cannot find the database

The database should always be at the default path.

Check the directory at path: $InstallationDirectory$/DubitaC_Data/Database

Where $InstallationDirectory$ is the folder where the game is installed, the one containing the file DubitaC.exe.

In case the database is not in this path, a new empty one will be automatically created the next time DubitaC is run.

I accidentally deleted the database

Unfortunately, there is no redundancy measure for the database in the current version, this means that when the database is deleted, all of its information is lost.

Try to restore the database file in other ways.

1.3.3 Shared

I want to play DubitaC over the internet

The current version does not support over the internet connections.

Setup a LAN to start playing.

Red lines appeared at the bottom of the screen

The red lines represent errors or exceptions in the execution.

If it is possible to continue playing, **Press on the "Close" button that appeared alongside the red lines and continue playing**.

If it is impossible to continue playing:

- **Go to path: %AppData%/LocalLow/LorenzoTibaldi/DubitaC**
- **Copy the file called Player.log**
- **Contact a game maintainer and provide them the log file.**
- **Close the application.**

The executions are taking an infinite amount of time

If it became obvious that the game is not proceeding to the next scene, it is most likely not due to the executions continuing infinitely, but because one of the clients has disconnected during some critical networking moments.

The current version cannot recover from such a failure, we suggest to **close and reopen the application to start a new game session**.

DubitaC requires a stable local network connection to ensure a smooth game experience.

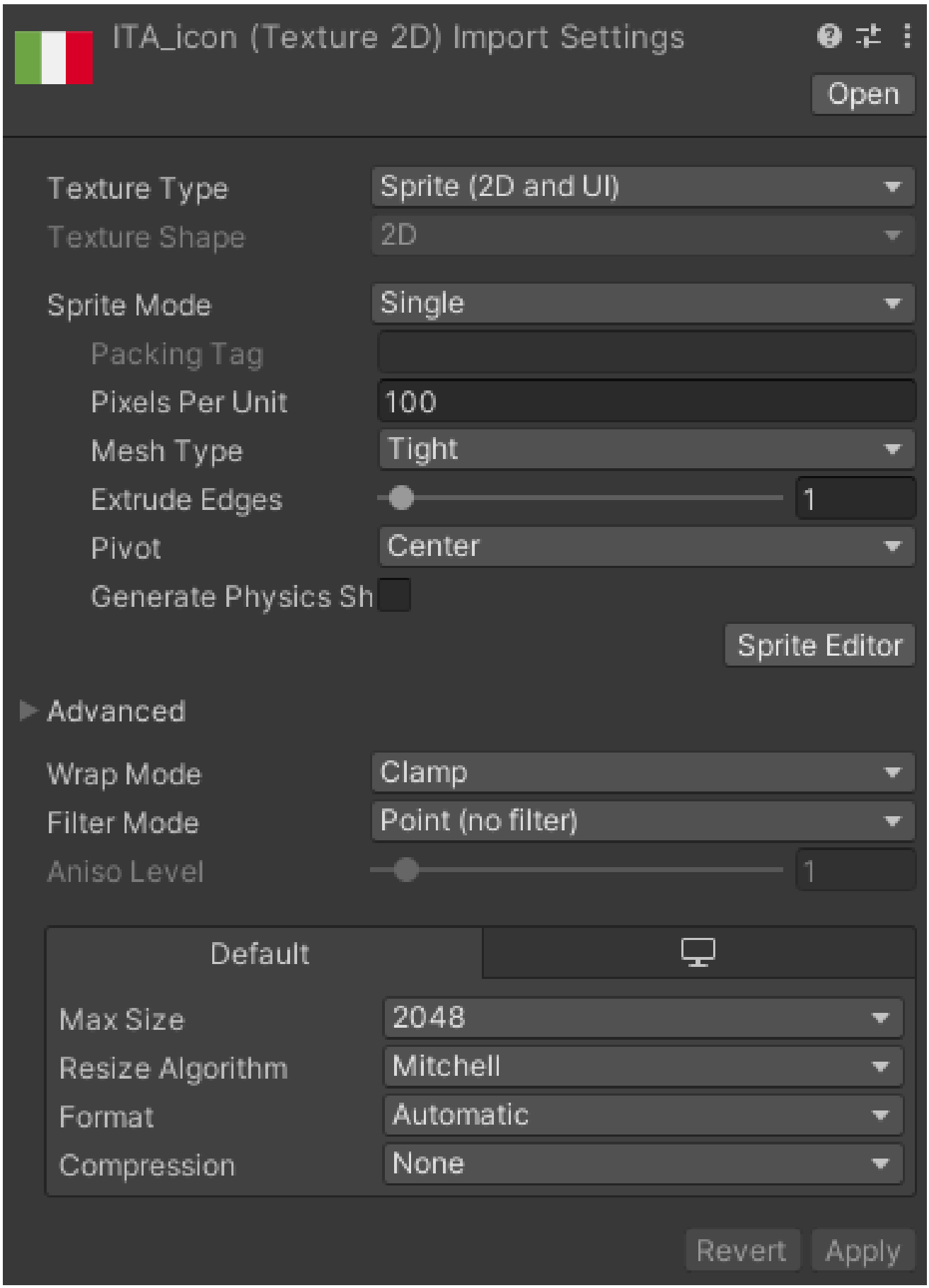

Figure 1.3: Settings to be used to convert an image into a sprite using the Unity inspector.

```
/// <summary>
/// Utility function to setup a <see cref="Process"/>, start it, wait for it to finish and dispose of it.
/// The function is 'async' because the execution of the command could take arbitrarely long.
/// The function returns a <see cref="Task"/> because we need the result of the execution.
/// </summary>
/// <param name="executionProgram">Name of the program to start.</param>
/// <param name="command">Command that will be given to the <paramref name="executionProgram"/> as argument.</param>
/// <returns>The result of the execution as it would have been printed to standard output.</returns>
private async Task<string> ExecuteProcess(string executionProgram, string command) {
    Process exeProcess = new Process();
    exeProcess.StartInfo.WorkingDirectory = Application.persistentDataPath + "/";
    exeProcess.StartInfo.CreateNoWindow = true;
    exeProcess.StartInfo.UseShellExecute = false;
    exeProcess.StartInfo.RedirectStandardOutput = true;

    exeProcess.StartInfo.FileName = executionProgram;
    exeProcess.StartInfo.Arguments = command;

    exeProcess.Start();

    string commandResult = await exeProcess.StandardOutput.ReadToEndAsync();

    exeProcess.WaitForExit();
    exeProcess.Dispose();

    return commandResult;
}
```

Figure 1.4: Asynchronous function used to spawn a terminal and execute arbitrary commands on it.

```
    /// <summary>
    /// Function to compile the user solution.
    /// The function is 'async' because the compilation takes some time (usually between 10 and 30 seconds).
    /// The function returns a <see cref="Task"/> because we need the result of the compilation.
    /// </summary>
    /// <param name="fileName">The name of the file to compile, with extension excluded.</param>
    /// <returns>The result of the compilation, if it is not empty something went wrong.</returns>
    public async Task<string> Compile(string fileName) {
        string executionProgram;
        string command;

#if UNITY_STANDALONE_LINUX
        executionProgram = "bash";
        command = "g++ -O3 -g0 -Werror -Wall -fuse-ld=lld catch_main.o " + fileName + ".cpp -o " + fileName + " 2>&1; exit";
#else
        executionProgram = "cmd.exe";
        command = "/C g++ -O3 -g0 -Werror -Wall -fuse-ld=lld catch_main.o " + fileName + ".cpp -o " + fileName + " 2>&1";
#endif

        return await ExecuteProcess(executionProgram, command);
    }
```

Figure 1.5: Example function that uses directives like "#if UNITY_STANDALONE_LINUX" to execute different commands based on the detected platform.