Spritesheet Saver

a Unity Plugin
Instruction Manual

Table of Contents

0. License and missuse warning....2
1. Introduction....3
2. Limitations:....3
3. Generating spritesheet from Unity 2D Mecanim animation....3
4. Additional Controls....5
5. Importing spritesheets back to unity....9

Made by:

0. License and missuse warning

This asset is covered under the publisher license (such as Unity Asset Store EULA). In unspecified cases or when publisher doesn't provide a proper generic license, following applies.

The software is provided 'As Is', without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, firness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

Buying this plugin grants you the non-exclusive, non-transferable, royalty-free right to use the tool for one user, with unlimited number of projects and devices. Redistribution, sublicensing, reverse-engineering of this package content, in part or in whole, with or without any modifications is prohibited without an explicit permission. Copying is granted for backup purposes as well as for multiple devices provided that all devices are managed by the user.

Please mind that while this software allows the user to replicate the work done by others it is strictly prohibited unless explicit permission is granted by the owner of original asset. Plagiarism is imoral and we will treat any attempt to use our software in such a way, as a violation of this license. Any violations will be further reported to the publisher.

1. Introduction

Mecanim animations allow Unity users to build complete animations without the need for any external tools. In addition the trigger system allows use of scripts, making it an ultimately flexible, and efficient mechanism for animation creating. However two problems appear when animations are being made in Unity:

1) animations containing many elements and custom curves are computationaly expensive. Essentially every animated object might need to store large amount of information for itself and its' children. This can be troublesome when performance is critical.
2) The animation cannot be reused outside without any tool being able to parse .anim and prefab files and import them as external packages. This essentially limits the reusability of an asset in other projects, such as graphics asset packs or game ports.

The simplest and commonly used method for preserving animation sequences, is to create a spritesheet animation, that is to dump specific animation frames as images. The generated spritesheet can then be imported using most of available animation/game development software (including Unity), increasing your work re-usability.

This plugin allows you to take existing animation clip and record it as a keyframe animation. The animation can be next saved as a .png file, ready to be re-used.

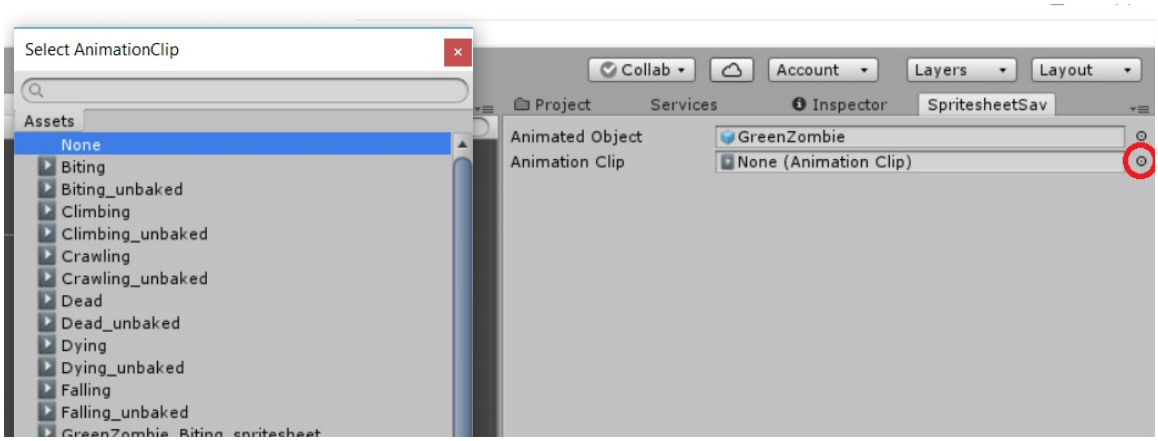
2. Limitations

1) The plugin is built using .Net Framework 3.5, and it requires a version of Unity capable of supporting this technology
2) Animations are recorded using the animation preview outside the 'Play' mode. This means that animation trigger events won't be triggered and therefore any change to the static game object will not take place. All animation changes have to be therefore controlled using animation curves.
3) Unity limits the size of Texture2D to some predefined values that might change in the future. This limits the possible output size for a single generated file. If the file size is too big for a spritesheet, keyframes have to be recorded to separate files and then merged in an external file if necessary.
4) Dense packing of keyframes is not allowed in current version. To keep things simple, all frames of a given animation have the same dimensions, and are distributed as a grid.

3. Generating spritesheet from Unity 2D Mecanim animation.

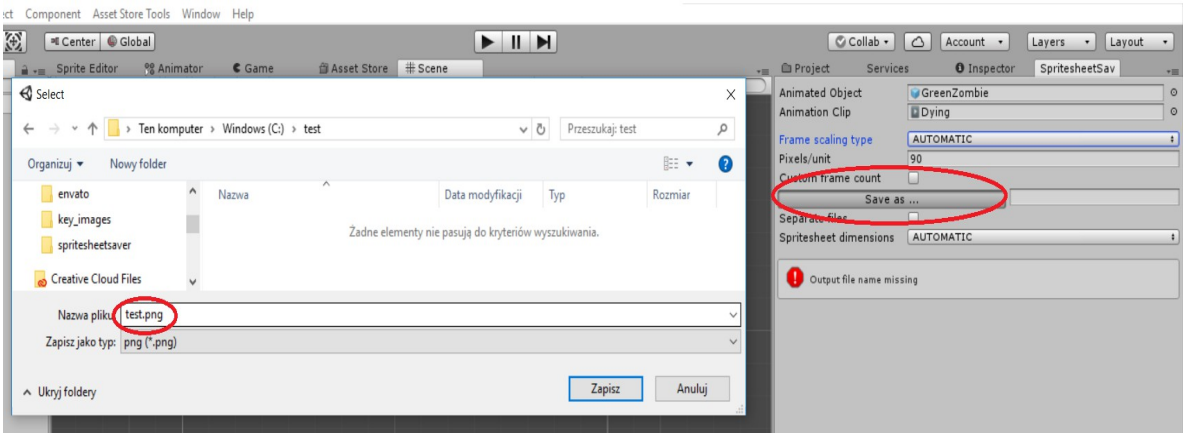
1. Create a new scene (**File->New Scene**)
   ****This plugin uses default camera and reconfigures in the process of spritesheet generation. It is advised not to use it in the production scenes.****

2. Open the plugin inspector (**Window->Spritesheet Saver**)
3. In the “SpritesheetSaver” window select object to be animated as well as a proper animation clip:

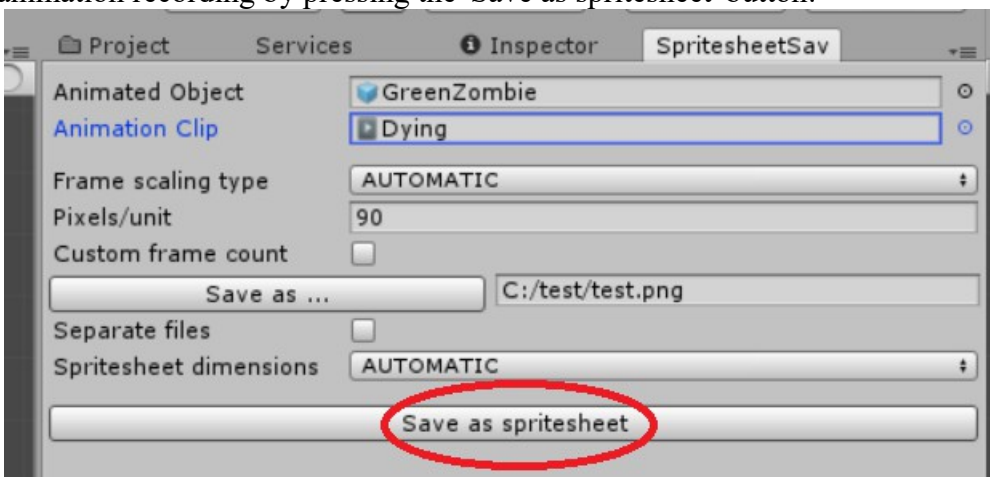


- For the 'Animated Object' field, either an existing scene object or a prefab can be selected.
- Make sure that the 'Animation Clip' is well defined for the previously selected object

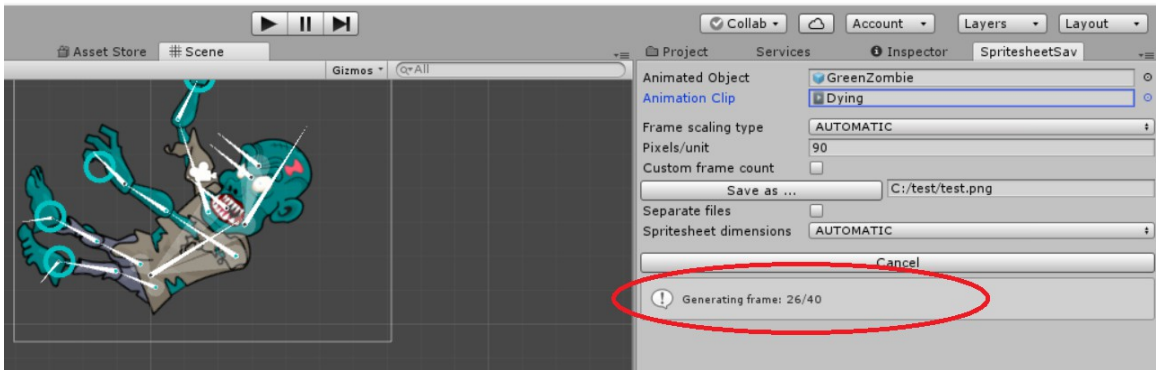
4. Select target filename by clicking the 'Save as...' button:



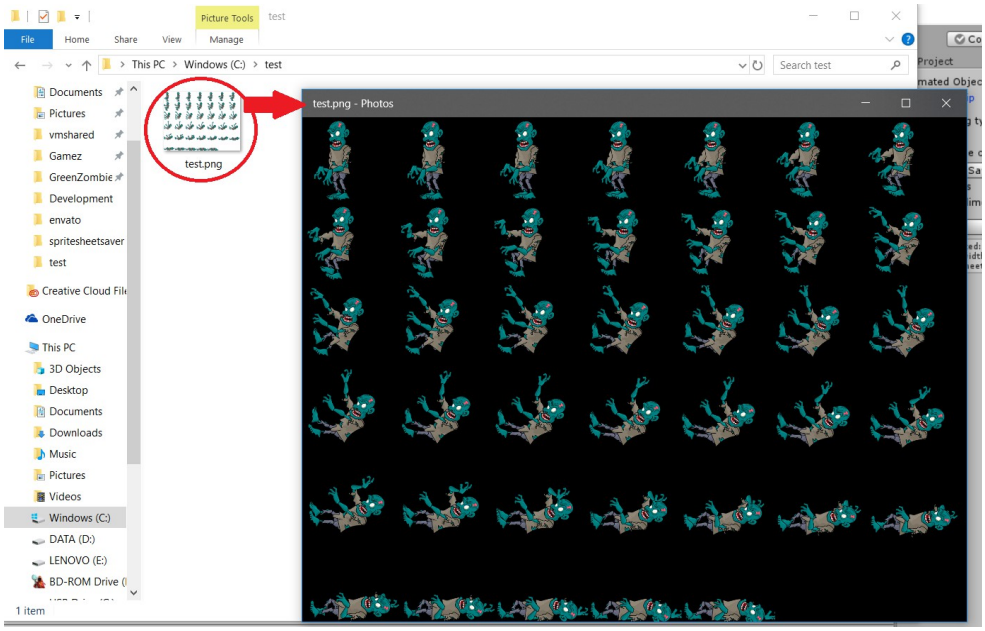
5. Start animation recording by pressing the 'Save as spritesheet' button:



6. Wait patiently until the animation records itself. The progress will be shown below the menu controls.



7. Inspect the output file(s) and make sure all frames were rendered correctly.



4. Additional Controls

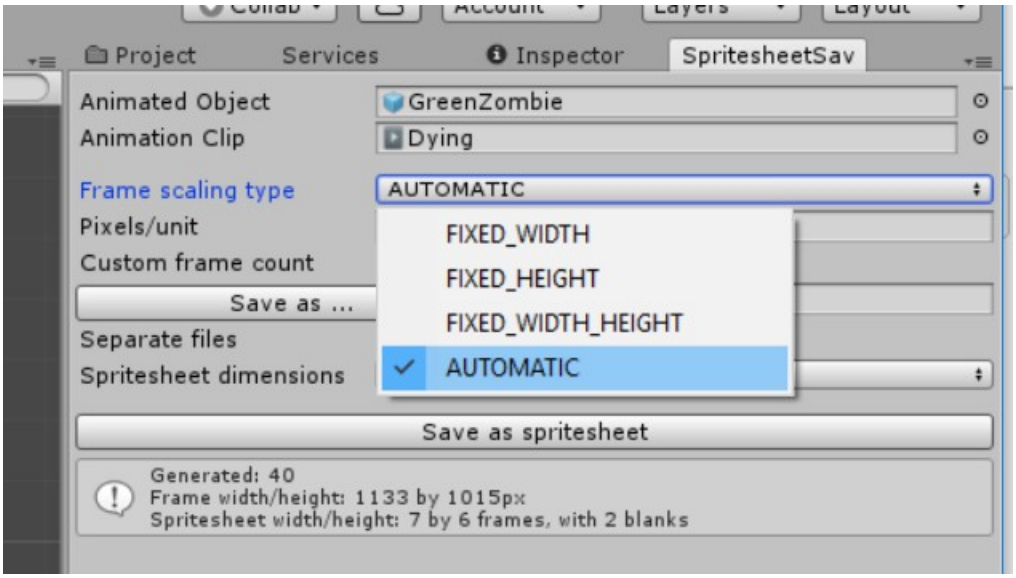
The plugin has some additional controls allowing you to customize the spritesheet creation. Following controls are available

4.1 Frame scaling type

Four options are available:

– FIXED WIDTH: each animation frame will have width (in pixels) selected by user. The height will be scaled retaining original sprite proportions

- FIXED HEIGHT: similar as above. Height defined by user, width calculated proportionally.
- FIXED WIDTH HEIGHT: both width and height are determined by the user. This can cause sprite to be stretched/compressed
- AUTOMATIC: the output dimensions will be determined automatically.



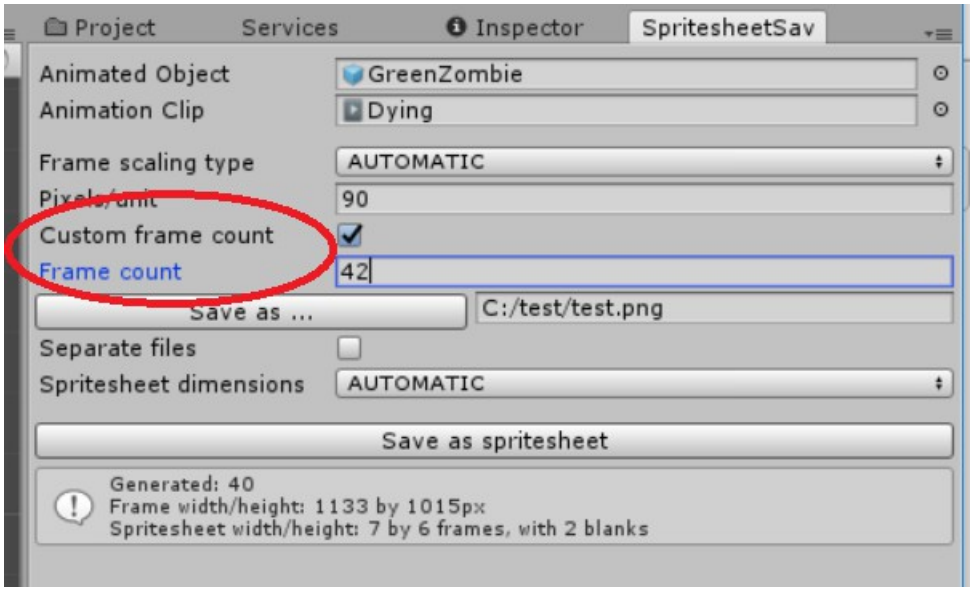
Regardless of option selected the specific values for frame width/height will be printed at the end of saving process.

4.2 Pixels/unit

This option is used for setting up camera. It defines the camera density as number of pixels per Unity coordinates unit. For instance if the value is set to 50, the sprite with width/height of 3x2 unity units, will be rendered using the camera of 150x50 pixels. If the **AUTOMATIC** frame scaling type is selected, this will be the dimension of each frame. For **FIXED** frame scaling types, the output frame will be downscaled or upscaled, which can possibly cause pixelation.

4.3 Custom frame count

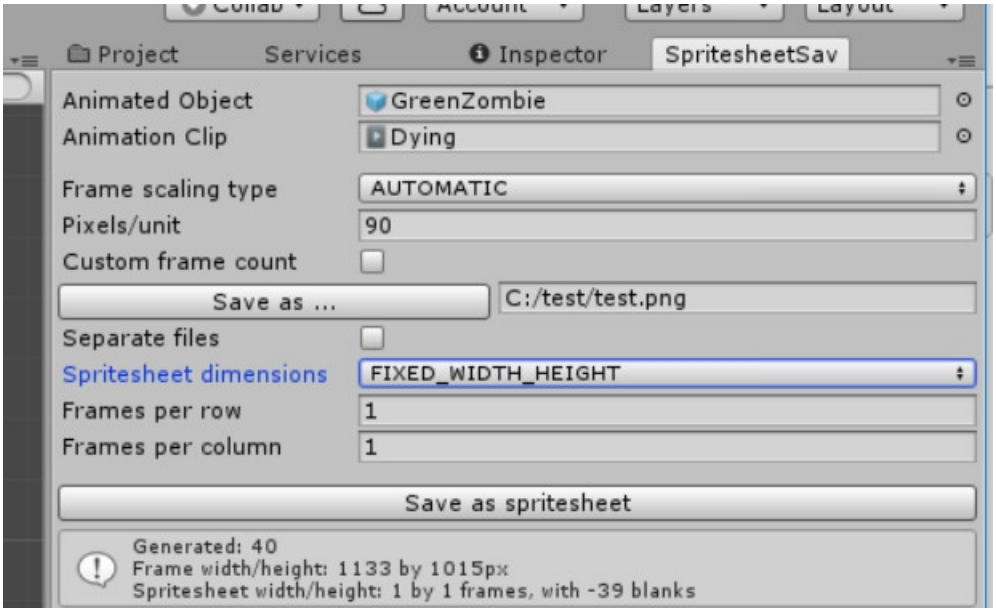
Ticking the 'Custom frame count' toggle will allow inputing a user-defined number of animation frames. If this function is disabled, then the plugin will determine the number of frames based on the animation clips' framerate and length.



4.4 Separate files

When ticked, the keyframe animation will be recorded with each keyframe in a separate file. The name of files will be determined based on already selected output file, with number of keyframe appended at the end of file.

4.5 Spritesheet dimensions



Four options are allowed:

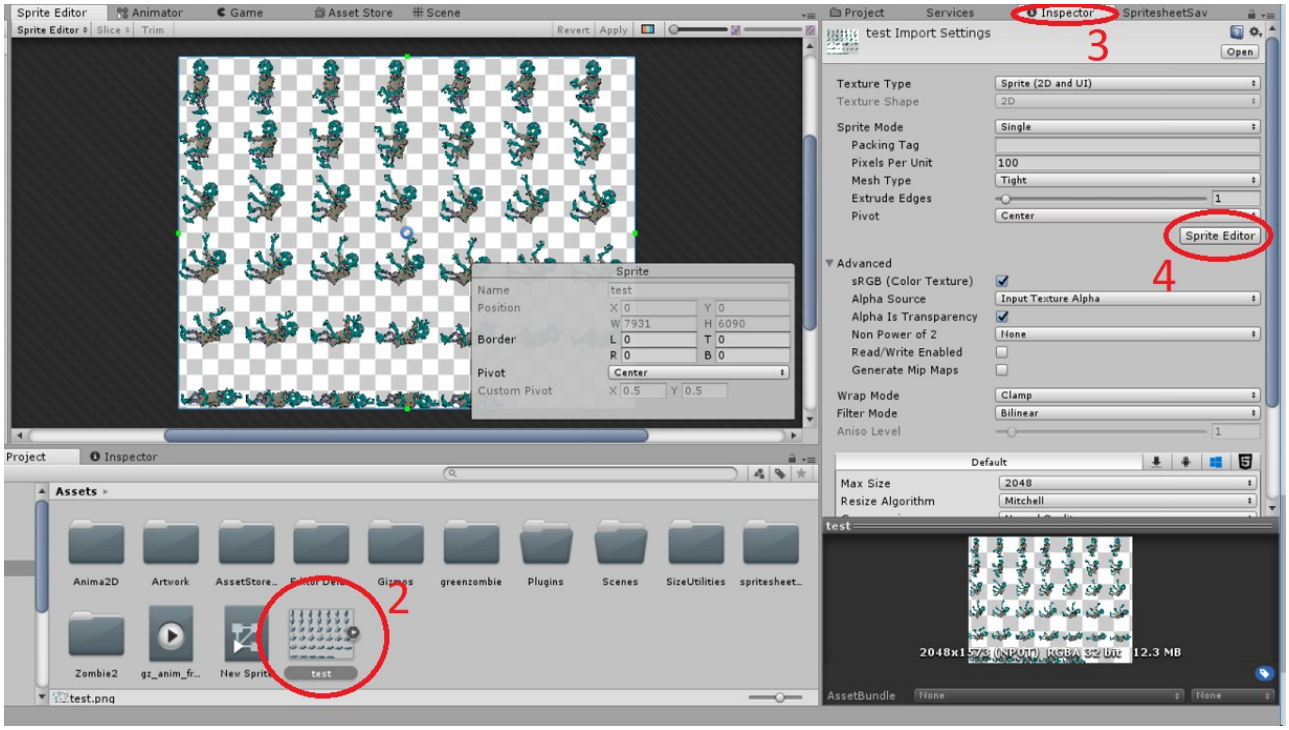
- FIXED WIDTH: the number of keyframes in a row has to be presented by the user. The number fo rows will be determined automatically.
- FIXED HEIGHT: the number of rows In a spritesheet file has to be presented by the user. The number of columns will be determined automatically.
- FIXED WIDTH/HEIGHT: both dimensions of keyframe grid are up to the user. If the number of frames is bigger than the size of grid, the excess frames will be dropped. If the number of frames is smaller, the grid will be filled-up with blanks.

- AUTOMATIC: allows the plugin to select the sprite dimensions on its' own. This will never drop any keyframes, but might result in some blanks.

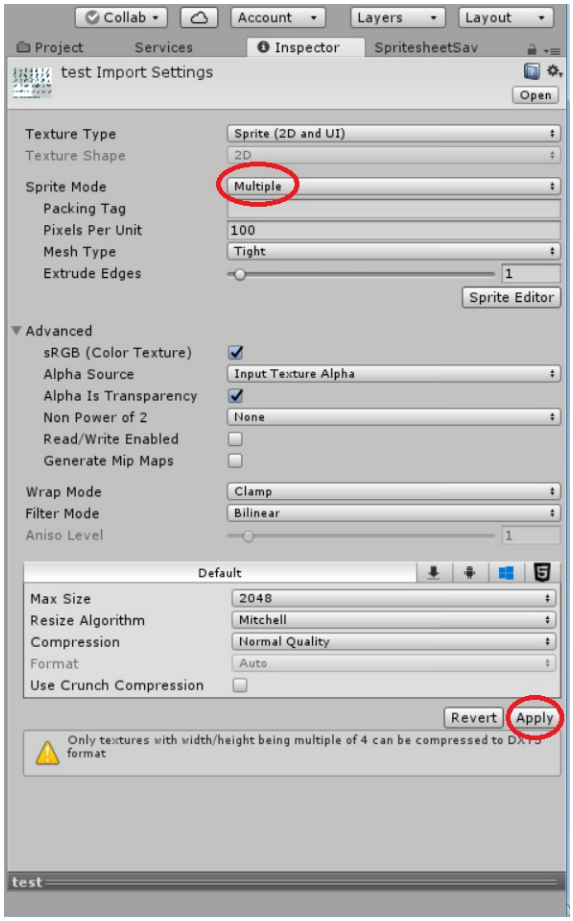
5. Importing spritesheets back to Unity

Importing a spritesheet to unity is pretty straightforward.

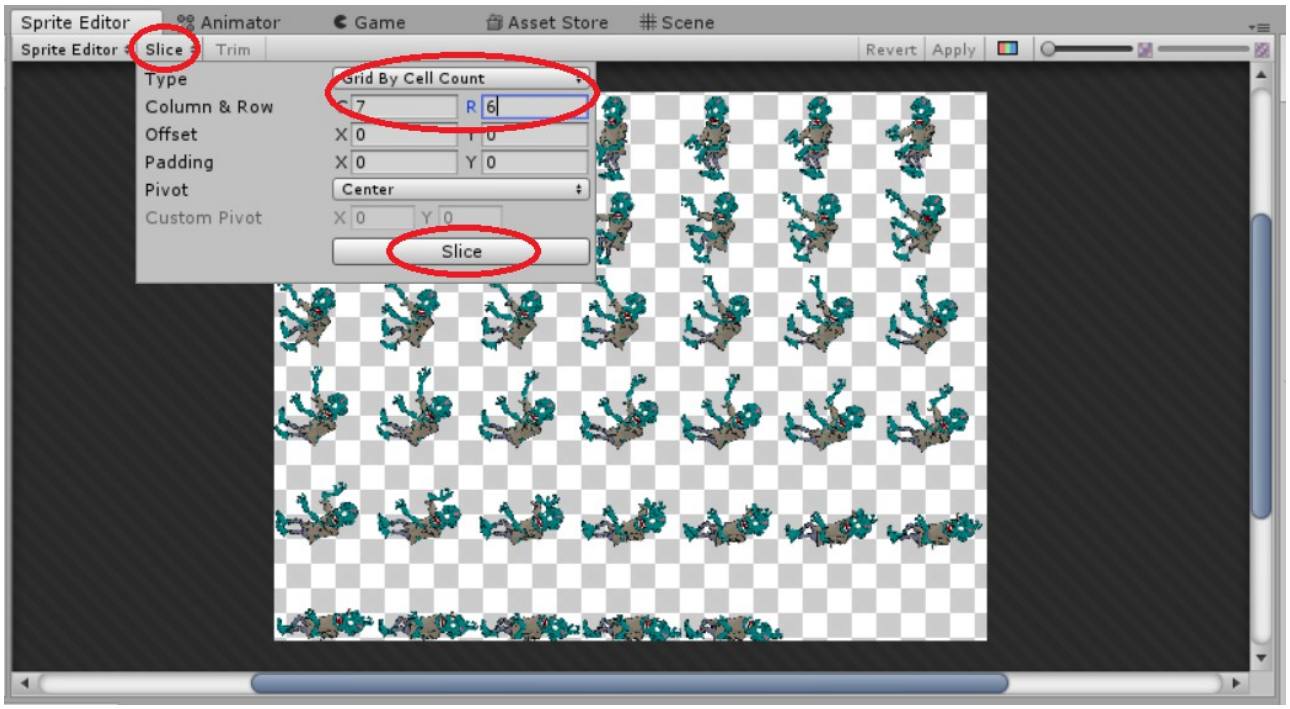
1. Copy the generated spritesheet to **Assets** folder of your project
2. Select the spritesheet in the 'Project' window
3. Open the 'Inspector' tab
4. Press the 'Sprite Editor window to open the 'Sprite Editor' tab



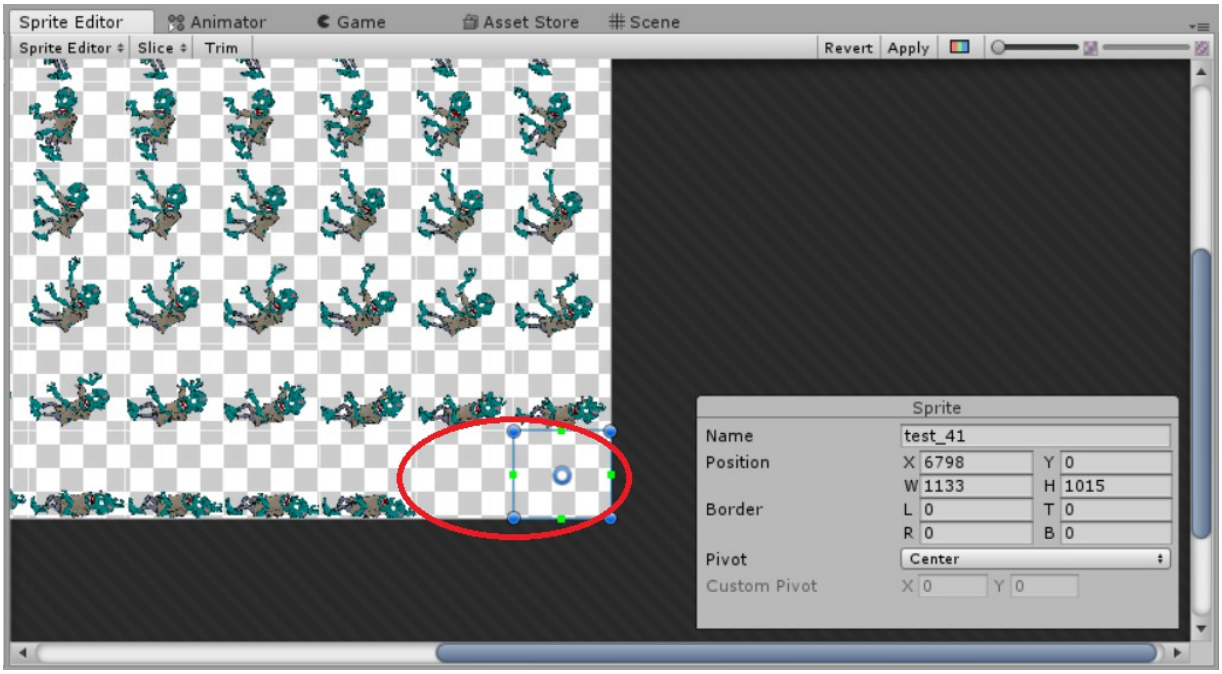
5. In the inspector change the 'Sprite Mode' option to 'Multiple'. Click the 'Apply' button at the bottom of Inspector.



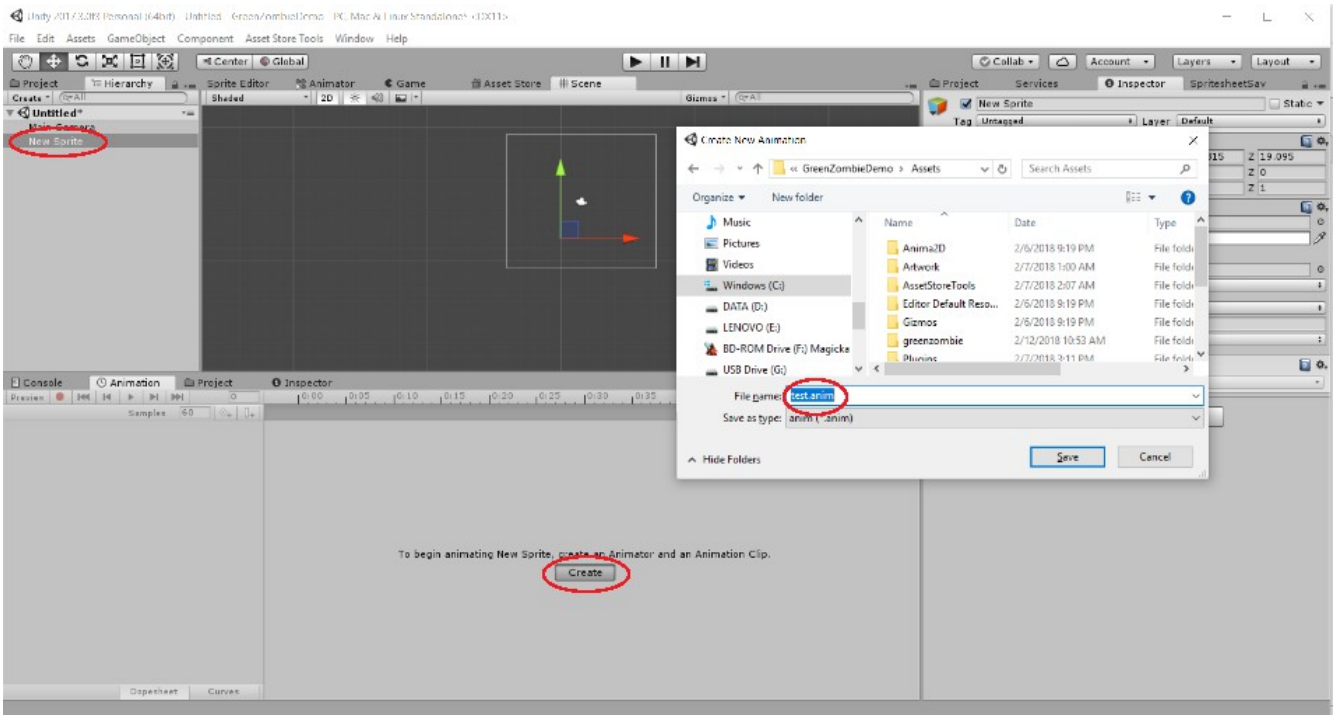
6. In the 'Sprite Editor' window select 'Slice'. Set the 'Type field' to 'Grid By Cell Count' and ender the sprite dimensions. Press the 'Slice' button to partition the grid.



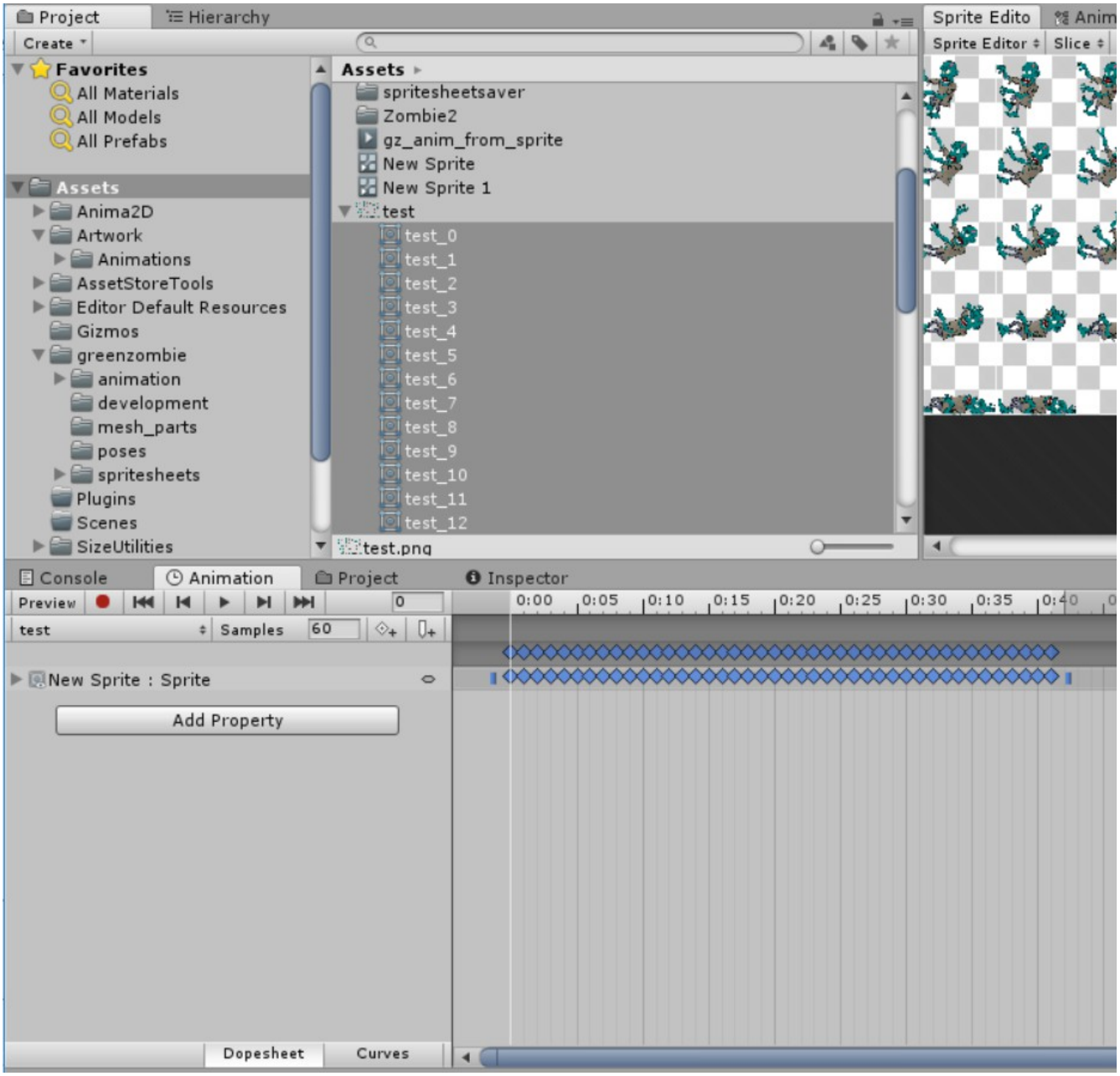
7. Select and delete (Edit->Delete**)** any blanks present in the spritesheet.



8. Press 'Apply' button in the 'Sprite Editor window' to save changes.
9. Instantiate new sprite object in the scene and create new sprite clip.



10. Navigate the 'Project' window and find the spritesheet file. It should now be possible to view specific frames. Select all frames (LMB + Shift) and drag them to the 'Animation' window's timeline. Adjust the length of the animation sequence to match the original.



Dumping a spritesheet might result in decreased image quality due to resolution change. Low number of frames might result in a jagged animation. At this point validation should be performed whether the reconstructed animation matches the quality requirements of the original.

6. Contact details

For any feature requests or feedback please contact us at enkstudio@wp.pl. For more of our work see our website: https://enkstudioblog.wordpress.com/.