Automated Regression Tests for Character Animation Systems

Hubert Mohr-Daurat
Creative Assembly
Hubert.Mohr-Daurat@creative-assembly.com

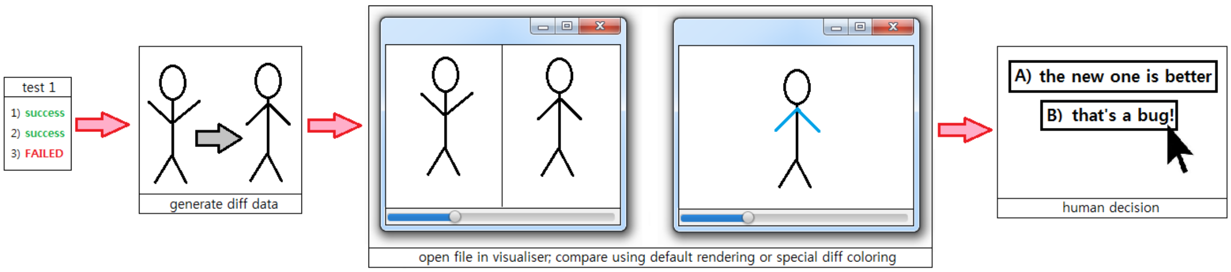

Figure 1: The regression test pipeline for character animation.

ABSTRACT

We present a process to verify code changes in an animation system by using regression tests which guarantee to cover every combination of animation features used in production. For this, we need to identify the untested combinations of animation features, create an immutable set of animation data which is representative of it, and when performing the tests, automatically compare the difference in the generated poses at each revision of the code.

CCS CONCEPTS

• **Computing methodologies** → **Procedural animation**; • **Software and its engineering** → **Empirical software validation**;

KEYWORDS

Character Animation, Regression Tests, Automation

ACM Reference format:
Hubert Mohr-Daurat. 2017. Automated Regression Tests for Character Animation Systems. In *Proceedings of SCA '17, Los Angeles, CA, USA, July 28-30, 2017,* 2 pages.
https://doi.org/10.1145/3099564.3106641

1 INTRODUCTION

Regression tests are commonly used in software production environment to identify as early as possible if the software still performs correctly after any change in the code or data influencing the core mechanisms.

However, it is more problematic when it comes to apply it for changes in an animation system. It is nevertheless desirable to have automated tests when doing changes in the code that could potentially affect a lot of existing animations. It makes the changes much safer. To state the problem to solve, we assume that we are in the case of character animation using skeletal hierarchy.

SCA '17, July 28-30, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5091-4/17/07.
https://doi.org/10.1145/3099564.3106641

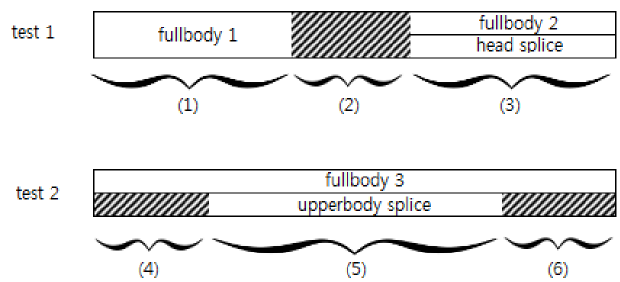

Figure 2: Two examples of tests (the 6 combinations are described in Table 2).

First, we need a method to evaluate the output of the animation system. It requires to work with animation data and to analyze the result by comparing the generated poses (joint transformations at each frame) between two versions of the system. Additional human evaluation need to be considered in the process to be able to discard negligible or desired modifications in the result.

Second, animation data used for evaluation cannot be the data used in production. This is because the data for the tests should be immutable, to guarantee that a difference in the pose is only due to a change in the animation system. It is therefore needed to create a specific set of animation data for the tests.

Third, this specific animation data set should cover every supported combination of animation features used in production. This is important to have a process to guarantee it since it is not straightforward to enumerate all the possibilities when combining various feature types: directional blending, transitions, body part splicing, bone retargeting, hands and feet IK, LookAt, etc. We will show that it is possible to identify the combinations used in production by marking the code with tags to generate unique identifiers for each combination.

2 OUR APPROACH

The process can be split in two parts: first, identifying the combination of animation features needed to be tested and creating animation data for the tests, and second, regression tests and output evaluation.

Table 1: Non-exhaustive (and simplified) list of animation features which can potentially be combined in the animation system, and their unique tag.

Feature	Unique Tag
Fullbody animation	F
Head animation	h
Upperbody animation	u
No animation	o
Transition	2
Body Part Splicing	s()

Table 2: Each tested combination for the two given examples (see Fig. 2), and their unique combination id to identify them at run-time.

	Description	Combination Id
(1)	Fullbody only	F
(2)	Transition from fullbody only to fullbody + head splice	F 2 F s(h)
(3)	Fullbody + head splice	F s(h)
(4)	Fullbody + blending in upperbody splice	F s(o 2 u)
(5)	Fullbody + upperbody splice	F s(u)
(6)	Fullbody + blending out upperbody splice	F s(u 2 o)

2.1 Creating animation data for the tests

The combination of animation features, like the ones listed in Table 1, need to be uniquely identified in the code. For this, we mark the code corresponding to a specific animation feature with a *unique tag*, represented by a symbol (letter or number). When a specific path is taken in the code, a *combination id* is generated depending on which features it went through and in which order. The tags can be chained or composed. Some examples of combinations are provided in Table 2.

Both production environment and regression tests can generate a list of all the combination ids they encounter, and then the two lists can be compared to know which ones are missing from the regression tests. The combination id can be decoded to generate a more readable description of the combination which need to be added to the test data. After this process of checking the error messages for all the missing ones and adding them, we can be sure to have set of tests which cover all the combinations used in production.

One interesting additional consequence of using this process, is the ability to detect at run-time any combination of features which are currently unsupported by the animation system (since the regression test would not exist in this case) without having to rely on visual examination.

2.2 Regression Tests

The regression tests themselves need to compare the difference between the previously generated poses and the newly generated ones. A method such as the one described in [Madges et al. 2017] can be used to compare the joint transformations at each frame, and create a visualization of it for human evaluation (see Fig. 1). The user can have the option to see it with coloration on the skin like in [Madges et al. 2017] or with standard skinning and lightning, and decide if this is an undesired effect caused by the change in the animation code, or if the result is acceptable (better or similar quality).

These regression tests can potentially be used as well in the case where the change in the animation system is intentional, to check globally which cases have been affected and to evaluate the improvement on the quality for all the existing animations used in production.

When the test result is discard by the user, either if the difference is considered negligible or acceptable (or even desired), the previous generated pose for this test is replaced by the new one, to be used as reference for future regression tests.

3 CONCLUSION

By using this method, first it allows to safely modify the code which affects the animation system behavior and rely on the regression tests to guarantee to have it working for any combination of the supported animation features. And it can also be used to warn the user when a specific combination of features introduced in production is not yet supported by the system.

This process has been considered to work with code changes in the animation system, but it could as well be extended for other types of changes which could potentially affect the quality of the animations. It would be particularly useful to apply a similar process when using motion graphs, to be sure to keep the graph stable and not break some states when adjusting some other states. Example-based motion synthesis systems could also benefit from regression tests, such as for evaluating the quality of the synthesised animations when modifying the motion capture data fed to the system and when adjusting the parameters used to analyse it.

REFERENCES

George Madges, Idris Miles, and Eike Falk Anderson. 2017. AnimDiff: Comparing 3D Animations for Revision Control. In *EG 2017 - Short Papers*, Adrien Peytavie and Carles Bosch (Eds.). The Eurographics Association. https://doi.org/10.2312/egsh.20171007