What are the difficulties of implementing algorthims that proceduraly generate game maps, in a highly coupled system?

COMP160- Software Engineering

1607804

March 20, 2017

1 Introduction

The implementation of new functionality to any exisiting system comes with complications. The one complication that the author wishes to focus on this essay is coupling. In the following essay, the author aims to provide insight on the difficulties of adding new procedural content generation(PCG) algorithms for the generation of maps. A description of problems that may arise from high interdependence will allow the reader to notice if any of these symptoms are apparent in any work they have done or are doing. In the game industry, working on a fully developed game system is common practice. In these cases, there may be little to no structure. By extrapolating research theory and case studies from the software engineering industry, the author will suggest stratergies to keep the integrity of your data when implementing PCG algorithms. In this essay, we define 'game maps' as the physical environment the player has to transverse whilst playing the game. Level design can be a tedious cycle of designing and playtesting to achieve the 'perfect' level. A level which is challenging enough for the player to enjoy but not too challenging it discourages the player [1]. This process is not only time-consuming but also expensive. One way to keep the players engaged but also to save resources is to automate this process through the use of PCG.

2 Discussion

In this essay, the content being procedurally generated is limited to world static objects. This includes but is not limited to Environment layout, Spawning locations, static enemies (turrets for example). When attempting to generate a map one needs to be aware of any circular dependencies about the object being called. Calling an object in a way so that one of the of objects cannot complete may result "in a deadlock or data loss" [2]. When implementing an algorithm that uses modules that have such dependencies special care must be taken to call them in a way that they will be successfully completed, or that if a deadlock does occur there are measures to solve them.(Needs case study or another such reference).

When choosing the objects that will make up the generated map, consider carefully if any of them contain traces of content coupling. Content coupling "is considered the worst type of coupling" [3, p. 95]. This occurs when a private method in one class can be invoked from another class. This means that not only is the code difficult to understand but also the one can not be changed independently of the other. When this is applied to map generation the results may be unpredictable. This high interdependence typically only occurs in dynamic objects, objects that need to respond to others and alter their behaviour on the behaviour of other world objects. By limiting yourself to using only static objects in your generation algorithm you can avoid behavioural uncertainty. Using a set of 'building blocks like sets of manually designed rooms to populate your map will result in a map that will look different with each generation but will fundamentally be the same. One can observe this approach in the games like diablo'[4].

There are various different metrics used to measure the amount of coupling in a program. One such system is called ”Coupling between objects” [5] this metric is quite a simple one. It is a count of all couples related to other classes not including through inheritance. If the CBO values for a class is high ”the reusability of a class will decrease” [6, p. 468]. Using classes with a high CBO value could prove to be very damaging to the state of any map generated, (Are Examples from my own work acceptable?).

.

3 Conclusion

Write your conclusion here. The conclusion should do more than summarise the essay, making clear the contribution of the work and highlighting key points, limitations, and outstanding questions. It should not introduce any new content or information.

References

[1] F. Kayali and J. Schuh, “Retro evolved: Level design practice exemplified by the contemporary retro game,” in *Proceedings of DiGRA 2011 Conference: Think Design Play*, 2011.

[2] S. Nair and R. Jetley, “Solving circular dependencies in industrial automation programs,” in *Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on*. IEEE, 2016, pp. 397–404.

[3] L. C. Briand, J. W. Daly, and J. K. Wust, “A unified framework for coupling measurement in object-oriented systems,” *IEEE Transactions on software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[4] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, “Procedural content generation: Goals, challenges and actionable

steps," in *Dagstuhl Follow-Ups*, vol. 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[5] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[6] N. P. S. Rathore and R. Gupta, "A novel coupling metrics measure difference between inheritance and interface to find better oop paradigm using c," in *Information and Communication Technologies (WICT), 2011 World Congress on.* IEEE, 2011, pp. 467–472.