**Impacts of FSM (Finite State Machine) In an ECS (Entity Component System)**

Executive Summary

In this article I will go over the negative and positive impacts of adopting the ECS and the coupling of FSM into the ECS pattern. I will cover topics such as **Migration of Code, Complexity of Code, Costs of Re-Training Developers** to showcase these impacts.

Associated costs

If we consider a standard monthly salary of a senior game developer at **$9000** and if the amount of developers working are 50. If a video game company wanted to make a game, with a complex ECS that handles memory management and component filtering with an intricate FSM coupled into the ECS paradigm. The time it will take to develop an ECS that is coupled together with an FSM, that is easy to understand for other developers and follows software practices such as the SOLID principles, is hard to estimate as a lot of work will be done in the designing of the ECS paradigm and the coupling of FSM with ECS paradigm. A best estimate of the time to take, would be around 2 months to design and 4 Months to actually develop this. If we take our standard salary into account and 6 months total development time. The costs of developing this for 6 months for one developer is ($9000 \* 6) **$54,000** and for 50 developers the cost will be (54,000 \* 50) **$2,700,000.** This has a very big impact, as the company might not have the capital to spend on 6 months of development time on something that the end user might not notice (They might want to put that capital into the advertising or art department that will make their game stand out more).

OO (Object-Orientated)

For small-scale games/projects OO can be very useful to create these games/projects. This is due to the fact that OO is very fast to develop in. This will reduce the cost of the production of small-scale games and less overhead for game designer’s, which will leave more capital to use in other departments. Even in AAA games designers and developer will work to create early prototypes of the game they want, and this could be done in OO to test out the fundamental game mechanics and make sure that the game they are making is immersive to the user. But this will fall short when you want to develop big scale AAA games, which will lead to large inheritance trees that will cause problems down the road in production. These problems would be issues where certain classes inherit in such a way that could lead to a problem known as the “Diamond Problem”. These issues can be solved if you use the ECS paradigm, that will get rid of inheritance all together and will make all the code decoupled leading to a cleaner codebase.

ECS And FSM Coupled Together

The idea of the ECS paradigm is to completely get rid of polymorphism and inheritance tress that will lead away from OO and into DDD (Data-Driven-Development). The reason why ECS goes this route is that OO does not scale easily into AAA games and will lead to problems that developers will need to get around. ECS is a very different way of coding. Instead of thinking characters/npc’s are objects, in ECS we think of them as data. This type of programming is hard to understand and even harder to implement, as ECS doesn’t use polymorphism and the developers will need to create an ECS system that is easy to develop in for other developers. This will lead to impact, where the developers will need to design out the entire ECS system and make sure that they have good memory management and really concrete component filtering, that will make it so that it’s really easy for other developers to create/use different components together.

In OO it is rather easy to implement an FSM into a game, as you can use polymorphism. All the states can inherit from a base state. This will lead to a problem where I have mentioned before, to a large inheritance tree. As I said before ECS doesn’t use polymorphism at all, so coupling together a FSM without inheritance will be very hard and time consuming. The biggest impact of this will be that it will take out time in development the game you actually want to create and into a concrete implementation of ECS with the FSM coupled together.

Summary

In conclusion, ECS with FSM coupled together, will lead to a better development environment, where the developers will be able to create and re-use different components without the risks that OO brings such as the “Diamond problem” and large inheritance tress.

Recommendations

In my opinion I would recommend companies that are developing big scale AAA games that have already developed a multitude of games with their own custom engine to integrate ECS with FSM coupled together. The reason for this, is by developing this paradigm, you will create a pattern that any game can use and will cause less overhead while you are creating multiple different games.

If you are a small-scale studio that wants to enter the game development industry. I would recommend the use of OO to get started, as you can implement prototypes and games in general pretty fast with OO whereas with ECS, there will be a big overhead on the initial development of the ECS paradigm that small-scale studios won’t have the capital and time for. But over time, after a small-scale studio as created a video game that the public audience like. I would recommend to create an ECS to make development easier on the developers.