1 Overview

MFSM or ~~Male-Female-Sadi-Masochi...~~ the Meta Finite State Machine is an attempt to answer two questions within the bounds of modern processing capabilities.

> How can we simulate a world of NPC's wherein organic interactions take place regardless of the player's input? *(i.e. The world is dynamic)*

–and

> How can we do so while acknowledging the limits placed on us by the ordinary computer?

It is not the intention of the current document to ask *why* but *how*. And the *how* is directly connected to both of the questions above.

2 Warnings to the Starry Eye'd

Dreaming is the birth child of genius or insanity. In many cases, the desire to produce something beyond how something is ordinarily done is forged by a whimsical imagination of how things *could* be done. But there is a difference between imagining the full extent of something, and imagining a different type of the same class. In these terms, it seems many projects are fraught with invalid starting points.

Consider any type of project as a house built from standard plywood and stucco. The dreamer, imagines he will build the Sears Tower and call it a house; even though it typically ends up looking like a mangled pile of steel beams. We, for this game, only want to build a house out of brick. That is to say that your design must be equivalent to your ability to implement. This immedietely precludes us from notions such as, *we'll just build a singularity!* Or, this will be true AI instead of that *crummy numerical nonsense!*[1]

This is all to say quite simply that the quality of the game does not lie in the AI itself, but if the design and supplementary parts concatenate into a finale which produces the intended results.

Put simply what do we want? We want–

> *The player to feel actual consequence for his actions.*

Which does not require some stupendeous AI that would make us far richer by turning it onto the NYSE. It only requires that we present a world *dynamic* enough to produce events we never intended to happen. Let us consider and introduce very briefly the notion of emergence, and how modern games might handle this sort of system.

3 Case Study: Spec Ops: The Line

Spec Ops: The Line *(Referred to as SOTL onwards)* was released in *2012* on several platforms. From my ancedotal recollection, it didn't seem to make many waves other than becoming a minor cult classic. You might wonder, why even bring up some strange game from almost a decade ago? It didn't do anything special. Wrapped in the guise of some psychological fruitcake of a narrative was a mediocore spray and pray romp across

[1] The 1970's saw a split in forms of AI with the symbolic and numeric approaches. Symbolic is technically the only approach which would yield a true AI, however, the 1980's AI winter put a stop to that because of retarded businessmen overselling shitty 4MHZ computers. That is not to say if you tried today you'd get some super AI. You wouldn't. Only a different type.

the deserts of who the fuck cares. That was SOTL. The claim for it's distinction lies in the proverbial onslaught of schizophrenic exposition, which if you look really deeply under the beautiful sand avalanches, might give you a minor jolt of excitement; if only because no other game dealt with the topics as well.

Yet, there is one part in the game that directly underscores my whole point. About halfway through the game you reach some cleft overlooking a city which happens to contain a few experimental artillery cannons. Napalm becomes the name of the game as the main character uses a computer to wreck complete firery destruction on the troops stationed down below. Within a few minutes, you find yourself walking through the city to get to your next objective, only to witness the still burning corpses of what looks like innocent civilians. This is usually regarded as a powerful scene because it doesn't outright tell you of who you've harmed but lets you discover it for yourself.

However–

It really doesn't mean shit.

Why you ask? Because after playing through the game again, I reached the same point in the timeline and realized there was nothing I could do but set everyone on fire. The moment you experience this, you realize that your decisions never really mattered. And in fact, there was no decision at all. You simply followed the path the game designers had laid down, and experienced what they wanted you to experience. This is, in my opinion, beyond terrible game design. I am no longer playing a story through another character's eyes but watching a movie.

In oddly theological terms, SOTL represents aptly the same desert where **Manichæian Determinism** was founded. It's all predetermined. There is no actual logical consequence to your action. There is nothing you can do to change that. Even being aware of it does nothing but reward you with the incoming steel fist of fate. In a way, SOTL has two endings. There is one where you play through the game normally, and another where you uninstall the game when you reach the firery hailstorm point. Because in doing so, you have conquered all Persian Mystics, and resigned yourself to the best possible fate.

To stop playing.

And that is not what a game is supposed to be.

4 Case Study: Fallout 3

Since we've already arrived at the banquet of theology and game design philosophy, let us turn our eyes toward a new religion; the CHILDREN OF THE ATOM.

Fallout 3 is possibly the best example one can give when attempting to argue for emergence in modern games. It lives in the class of games like Oblivion, and KotOR. These games seem to offer us choice, and alleges that it will keep it's promise of making our choices matter. (Or at the very least making us feel like they mattered.)

To begin our journey we might remember the case of MEGATON, an odd ramshackle of a city with a nuclear bomb for a heart.

A distinctive quest early on begins in the bar of Megaton, and the introduction of a shady suited character. We are offered a decision between helping the man arm the nuclear bomb, and thus destroying the city, or we might double

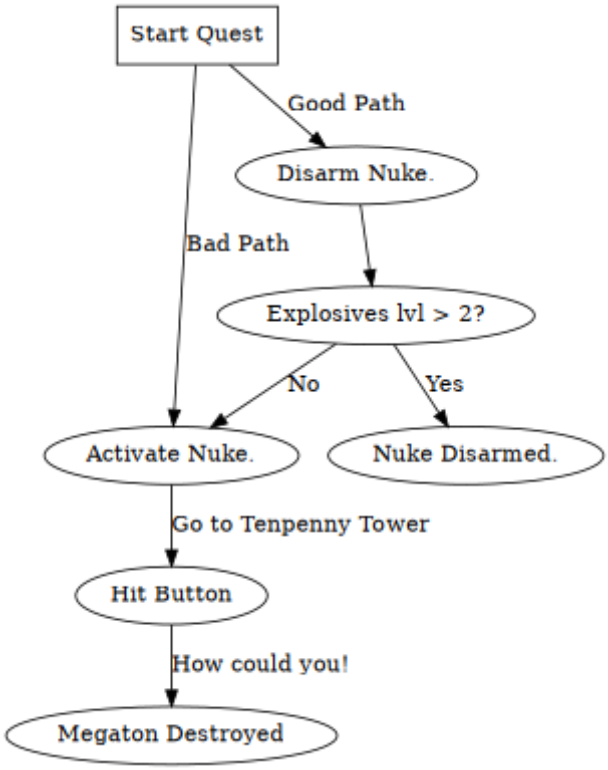

Figure 1: Just an average day.

cross him and disarm it completely, thus saving the city from a latent threat.

There are two massive problems with this style of gameplay. The first is that a consequence has been degraded down to a numerical conflict[2] and the second is that I was able to graph something that should be a huge moral question into a flowchart. The former is the *inner problem*; the latter the *overaching* one.

4.1 Numerical Degradation

Imagine just being propositioned by some 1950's businessman in a shoddy bar that you should arm a bomb in one of the last falling remanants of collective civilization. So you say,

> *Your outdated charm won't work on me!*
> *I'm going to save the city.*

–And you run to the bomb only to see a menu pop up...

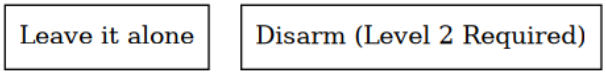

Figure 2: Numbers...My Arch Enemy.

–Oh no! You don't have level 2 in explosives!

You might think that the numbers in an RPG like this are supposed to be representative of a characters ability to perform an action; and they are. However, in this case a moral issue has been turned into a mechanical one. And here is how ridiculous it looks.

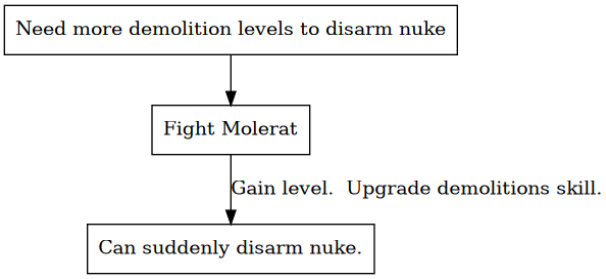

Figure 3: Dr. Molerat – Professor of Nuclear Engineering.

Does this really make a good story? I walk outside town, grind some levels and suddenly I can disarm a bomb? In my opinion, this completely destroys the story because at this point the player has recognized the numeric limitation placed on him. He is now operating in the **meta** to work around the game and reach his end result.

[2] *Explosives level high enough?*

What would be better? (but not the best)

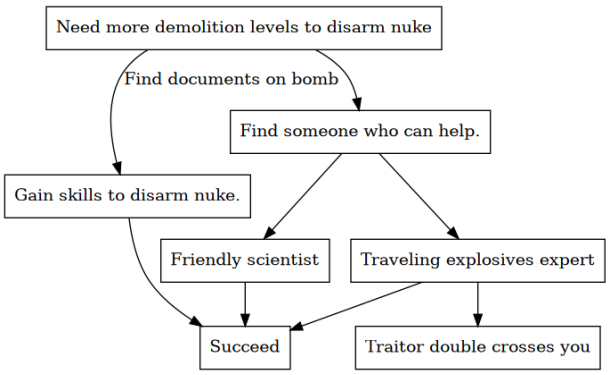

Figure 4: Fix it with mods?

This would be a far better solution to the current problem. It involves acknowledging the logical consequence of what the player has chosen beforehand[3]. It lets the player deal with *organic* looking help. For instance, the traveling expert should not always show up; nor does he always actually help.

Of course, this is much more work to make. But that is not because it is out of the developers reach, but rather the consequence of an intention design decision. There are no randomly traveling explosives experts. There is no scientist. The path laid down is one of using the numerical system to reach a predetermined goal. Which brings us to our second problem...

4.2 Determinism Again

Figure 1's ability to be graphed out is one thing, but another is the fact that this graph *will always* take place in the game. Every single player will be thrust into the same flowchart, given a meager chance to make a *decision* and then told that the game world reacted to their choice.

[3]He didn't choose explosives knowledge

Speaking philosophically again, this is just **Compatibilism**, which is another form of determinism. You are lead to believe you have a choice, and you do, but only if it falls in line with the pre-determined choices available.[4] And really there's no need to go on explaining this beyond giving a final example.

Perhaps, you should reconsider your design if I could literally print out an SAT style multiple choice questionare with all the *choices* that can be made in your game, let people fill in the bubbles, and tell them they've just experienced true consequence to their actions.

This is not what the height of games should be. It is a direction leading only to a proto-regression of player involvement. It would seem that the only answer to this problem lies once again in a smattering of philosophy

and drunken dwarves.

5 Case Study: Dwarf Fortress

If you were to ask any reasonable person on the street–

> *Which game will let me start an accidental goblin genocide because one dwarf couldn't hold his liquor?*

They would probably respond with–

> *Sir, I'm calling the police.*

But that's just because their naivety won't let them look past the distinct smell of craft IPA's and Cheezit's adoring your crusty unkempt beard in order to see the glorious story telling of **Dwarf Fortress**.

[4]This is still better than SOTL's lack of any decisions.

The greatest immediete property of Dwarf Fortress is that I can't actually graph it. Any of it. I could graph a *possible* event that *could* happen, but reproducing it would border on unlikely and isn't anywhere near the point of playing a game with true emergence.

This works because all the systems in the game fit together in ways the developer never intended, and in fact, he probably didn't *intend* to make you experience anything. You experience what the world happens to do, and you react to it. The world reacts to you, and you to it. If you were to leave the keyboard, the world would keep on spinning in abject loneliness. And you would probably lose.[5]

One realization from this, is that the game is not necessarily *designed for the player*, the world is designed to be a world, and the player gets to interact with it. In this way, the player's stories truly become *their* stories. The events and decisions become *their* decisions. A simple glance online will show hordes of people telling stories about what happened in their particular game of *Dwarf Fortress* and others intently listening. This simply doesn't happen in other games because everyone knows that their decisions aren't necessarily special.

It's the quiz show equivalent of–

Did you pick A or B?

I picked A.

Oh cool...me too.

–And that's pretty much the end of the conversation. Dwarf Fortress opposes this by throwing determinism out the window. Dwarf fortress is about everyone's favorite question. **Free Will.**

[5]But who really knows.

To demonstrate let me show a *possible* chain of events that could happen in a random Dwarf Fortress game.

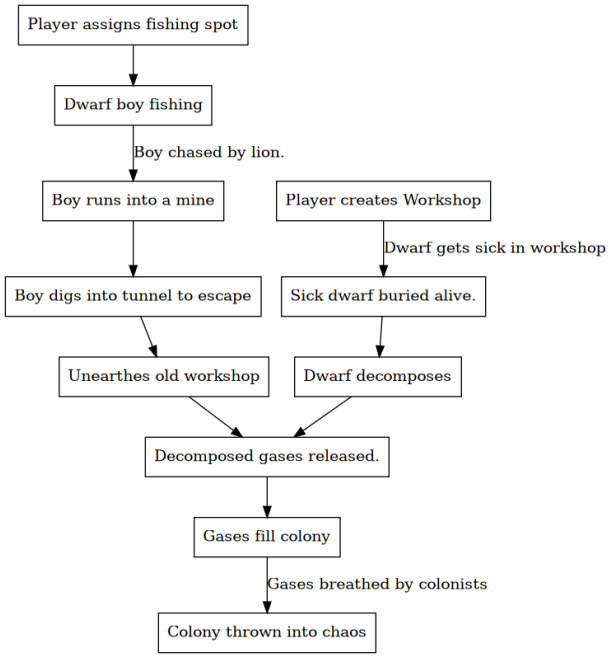

Figure 5: This is why we need Bernie.

None of these events are scripted. There is simply a system for NPC's to perform actions, a system for how a dead NPC changes, a system for an NPC running from a threat, and even one for how any type of gas might spread.

All of these systems combine to form the emergent gameplay. The player faced real consequences for his actions. He should have changed the workshop conditions, or picked a different fishing spot, or watched out for something like this happening. He didn't, because this was one small part of the whole colony he was dealing with. The world continued to react and act despite his involvement, and now a story has been created from the result.

The player, upon death of his colony, can now go online and tell everyone about how a fisher-boy caused his whole colony to die, and chances are that no one else will have experienced this quite the way the original player did. That is a true story, without intentionally telling a story, because the player feels a deep connection between what happened. He now knows that there is a logic to the world that he must understand, that is not necessarily numeric.

The game has changed from thinking–

I just need to level up this group...

Into something that forces the player to deal with many more aspects in the world. And those aspects come into being by the player's geniune choice. He is responsible and feels the consequences for the choice he's made, not because it was on some flowchart, but because he is playing in a real dynamic world.

Dwarf Fortress has many flaws, and it is not the game that is intended to be described here, but it does illustrate a new level to future gameplay. A level that will draw as much from the player as it gives him, and let him forge his own stories without forcing him onto a railroad.

6 MFSM Fundamentals

MFSM is a potential answer to the propogation of a dynamic world. We have already seen types of worlds, but now we must ask how we can create one ourself. In some ways, the idea of how to make an AI that can handle this is closely coupled to the design we wish to uphold. For the present, we will only discuss the theoretical implementation of a system that can lay the foundation for building up whatever design we intend on.

MFSM operates off of two novel concepts that grant it the **M**eta in it's title. But before we come to that, a simple overview of how AI might work in a modern game.

6.1 FSM Fundamentals

The field of **Finite State Machines** is a rather complex category, and I surely don't claim to have some great knowledge of the particular field. However, it is used consistently in video games if the developer knows that or not. A FSM can be viewed from a super basic level as something like a closure, or a tiny computer operating seperately from the main process. The emphasis is on the *finite* portion, wherein the state machine is only capable of expressing a finite number of internal states.

Here is how one might look in basic terms for the case of a simple NPC guard–

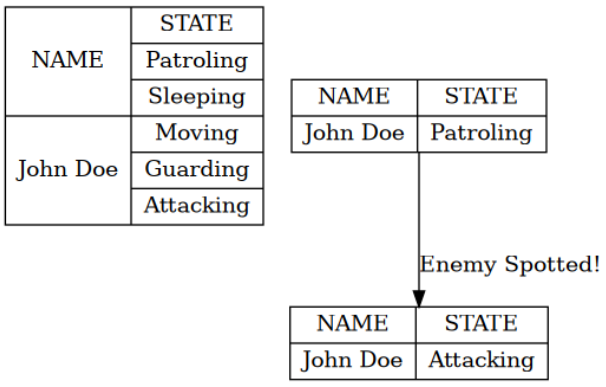

Figure 6: Keep these peasants in line!

A washing machine is a good example of a simple FSM. It doesn't make for a good game (unless...) but it's a state machine that has 5 different states. Any FSM will use some sort of internal notifier that causes it to change to a

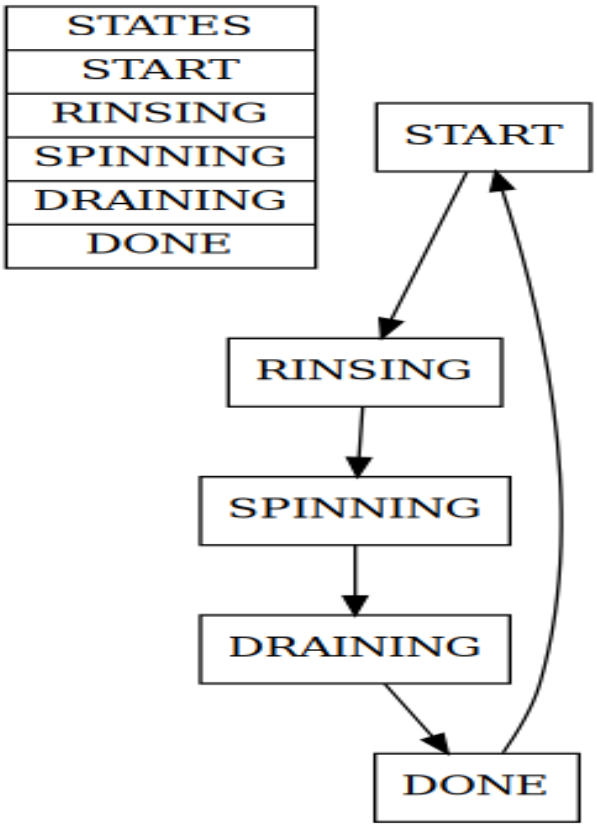

Figure 7: 0 customers served this week.

different state; a washing machine would use a timer.

Games do this much like Figure 6 with the guard. Most enemies are running an internal **FSM** that responds to input from the outside. For instance, a state machine may have an internal state describing the health points of an NPC and may shift states when these health points reach a certain low point. This is in itself a really basic form of A.I. in the player's eyes. He attacks something, it responds by running away when it's wounded.

Sometimes people use **OOP**[6] instead of an explicit **FSM**. The object holds an internal state, some update function is called every once in awhile that changes this state, and possible a certain value is reached that causes the object to take a new action. Maybe when a player attacks an object, the object checks if it needs to do something else depending on what happened to it.

The **FSM** can get extremely complex, but it is essentially a declaration that you know the potential states of a *thing*, and you would like to make a simple computer like object that can handle those states internally. You really don't need to know about them personally, the machine can handle them itself.

We can, and people have used these for many forms of game AI. It's easy to think about the potential states an enemy might have, and the neccessary reactions for those states. At the very least this concept is not really implementation dependant. You would not have to explicitly use an FSM library to use the concepts we're about to introduce.

6.2 The Problem

The Problem becomes–

> *Ok we have these states. But how does this help with a dynamic world? We can't enumerate all the possible states of a dynamic world! That'd take forever.*

We could handle the problem by releasing updates to the game that added more and more states until we finally reached something that looked dynamic enough to be called cool. And really it's a viable system–

[6] OOP, FSM, and Closures seem to be somewhat linked.

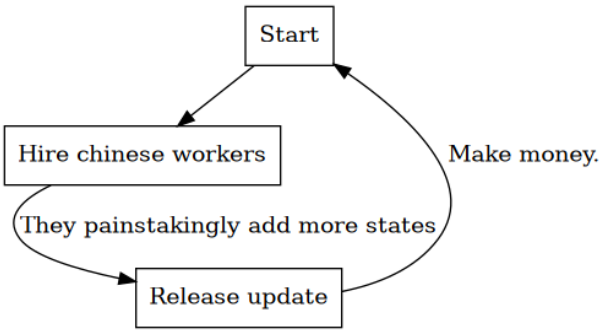

Figure 8: Capitalist Game Dev

However, the money spent evading international government regulations would probably outweight the cost effective savings of production. Perhaps, we can solve this another way? We have two problems. We need a more dynamic FSM, and we also need it to not tank our machine.

7 The Meta FSM

Suprisingly, a few months after coming up with a concept for how to solve this problem, I learned that the team on **Alien: Isolation** came very close to what I was thinking. The **Alien** in the game was perhaps the best part due to the perception that his A.I. was actually, fairly good. The player really didn't know what he was going to do, when he was going to come, when he'd leave, etc. And it worked out wonderfully. How did they do it?

They split his head in two.

That is there were two **FSM**'s running for the Alien. One of them was lower level and handled the *how do I do something* state. The other was higher level and determined the *what am I doing* state.

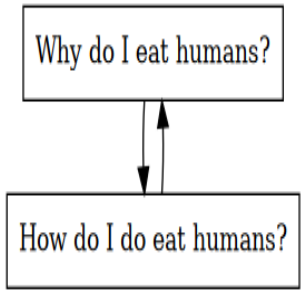

Figure 9: The Duality of Man

And this worked really well. The **Why FSM** directly guided the **How FSM**. This same splitting is essentially a singularly meta FSM. The most important aspect of this is that it subdivides the decision space into two seperate entities. The Game Developer does not have to enumerate all possible actions from the player, he simply has to give some personality to how the alien wants to react in general. When this **why** influences the **how** you end up with a dynamic looking AI.

This requires a split of the *mechanical* and *emotional* responses of the system. Let's return to the guard example to see how this might work if we gave him two brains.

Imagine our guard now has an emotional response to input that directly influences his mechanical response. We could say the guard's *emotional* response states are something like **Happy** or **Aggitated**. For *mechanical* responses we might say he is **Patroling** or **Sleeping**.

The point is not to say that by having the *emotional fsm* change state the lower level is guaranteed to change. It only means that some weight in a decision has been propogated to the lower level. The guard now has a higher chance to sleep on duty, not because of the random chance

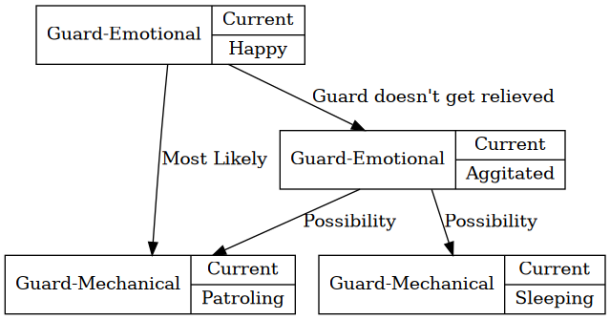

Figure 10: What is he even guarding?

in itself but because he was influenced by something else. This provides some loose form of *logical cause* in the AI.

The further point of this idea is, as mentioned, to subdivide the decision space. An emotion system for instance doesn't need to define what the mechanical system does. All any FSM needs to know is it's input, (in this case the input came from the mechanical system in the look action), it's states (happy or aggitated), and it's output. For instance, when the guard goes to sleep he does not wake up because of a timer. Instead, the mechanical system is outputting a positive result due to it's internal state (sleeping) which is fed into the emotional fsm. The emotional fsm eventually reaches a point where it's state flips from *aggitated* to *happy*, which feeds as output back to the mechanical system that is now influenced to change from *sleeping* to *patroling*.

This is a simple example but it showcases a couple points.

- Every FSM has knowledge of the inputs and outputs directly above or below it in the Meta-Tree.
- Since an FSM is aware of how another FSM will take input, adding a new state to that FSM is as simple as defining it in terms of how it will output it's own state in a way that influences the other FSM's state.
- Since state is atomically changed by subdividing the decision space, the entire group of FSM's can run outside the main thread continually and be queried while they're running.
- The lowest level of FSM's is the only one which is directly queried by the game engine. The game engine doesn't need to know the reasoning for why an NPC is doing something, only what he is doing.
- FSM's are pausable, but still queriable while paused.

This is the general idea of the **MFSM** system. The lowest mechanical level encapsulates general AI found in most games. Any Meta FSM level above this influences the level below and causes loosely defined logical causation from other inputs to propogate in the tree and change visible action.

But you're probably wondering...isn't this expensive as fuck? I mean, maybe it's useful to subdivide things like emotional reasoning, materialistic influence, and actual mechanical action, but unless you want less than 10 NPC's how is this not going to bring the computer to a crawl?

As stated before, the FSM tree does not need to give instantly precise answers to it's current state, but eventually precise. Which means that all FSM's can be running seperate from the game, the game simply queries the current state (which could be wrong at that exact moment because of multi-threading issues but that's ok because it'll get fixed by the next tick).

However, we can actually optimize this by utilizing spatial isolation and signal frequencies.

7.1 Spatial Isolation

The simplest way to put this is that every FSM can be encapsulated in an object that stores it's current position in the world. This can have an oc/cube-tree run over it to physical segregate the FSMs, which means only FSM's that are located in a physical proximity to the player will have priority to run. The rest of them will run, but at a much slower tick rate. Any FSM's in this proximity will actually use thread-locks to ensure the game is querying the precise value.

7.2 Signal Frequencies

Imagine a game about a group of gangs with different parts of a city. In the interests of a dynamic world it is entirely possible that a gang member on the other side of town (outside the prioximity of the player) is doing something important. We need him to be able to do this so that when the player arrives there later, a real change has occured. In this case, we simply assign an external weight to certain actions. A guard looking at something is much different than a gang memeber selling something. The latter is much more important than the former. In this way, each FSM can assign a weight to each of it's potential states. This weight is then allowed to be accessed externally.

After this, major quadrants of the FSM-tree are taken and have a DCT run over them to generate a frequency map of weights. If one quadrants has a spike in frequency that means that a particularly valuable action is happening within it. This quadrant is given a higher priority as the FSM-tree iterates over itself, which means someone across town will be able to perform their action if it's important enough without forcing every FSM to run because we have no idea what's important.

7.3 Conclusion

There is, of course, much more to the internal workings of an MFSM-tree, but this is an introductory and exploratory idea of what could be. The next section will deal with some of the theoretical underpinnings in why we might be able to represent a game state this way. The section after that will propose a simple scheme for how the gang game might be able to utilize it.

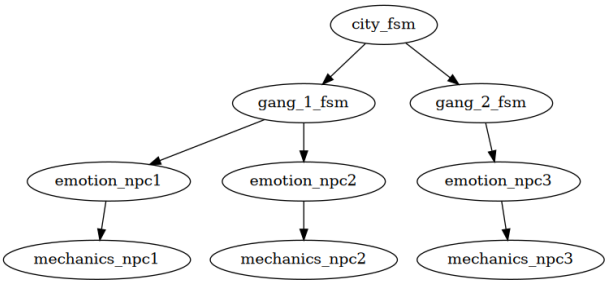

Figure 11: Simple Overview

The idea creates a certain complexity for how the systems have to be implemented but also seems to allow for flexibility in that one level of the system can be expanded and integrate perfectly into the rest of the tree. If emergent gameplay is decided by how well all of the systems fit together, than an unified approach to a subdivided AI decision process seems to fit into the goal. Especially, if such a tree can be queried by every other system in a run-time efficient manner due to spatial and frequency scaling.

8 FSM's and SFG's

9 A generic FSM-tree for the gang game