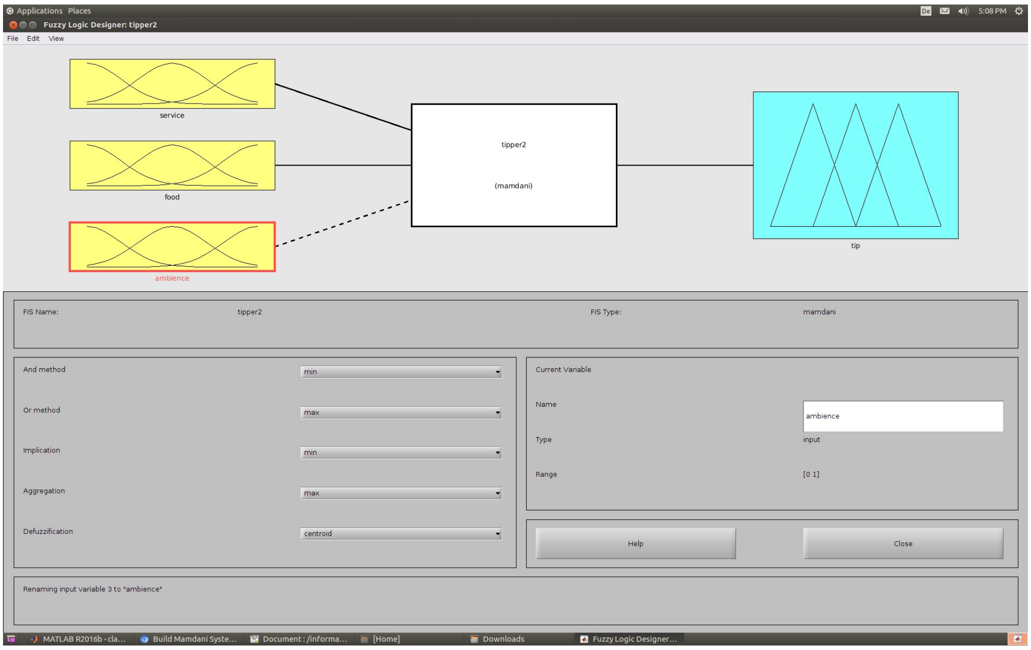
1

Since there is no explicit/specific question or work assignment in this task, I'll simply state a few observations for the given system *tipper* and then change certain traits of said system to alter those previous observations. This will hopefully show my understanding of fuzzy logic concepts and the usage of *MATLAB's* corresponding designer. With these basics out of the way, I'll keep the description of task 2's new system somewhat concise and to the point.

1.1 Selection of input and output

Currently, there are 2 input variables (service and food) which are used to determine the value of a single output variable (tip). This is quite the logical system since service and food are probably the biggest factors for most tippers out there. To add another reasonable input variable to this setup, one could imagine a tipper who also desires a nice ambience for a generous tip. Adding *ambience* results in the following system:



FIS Name: tipper2

FIS Type: mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

Current Variable

Name: ambience

Type: input

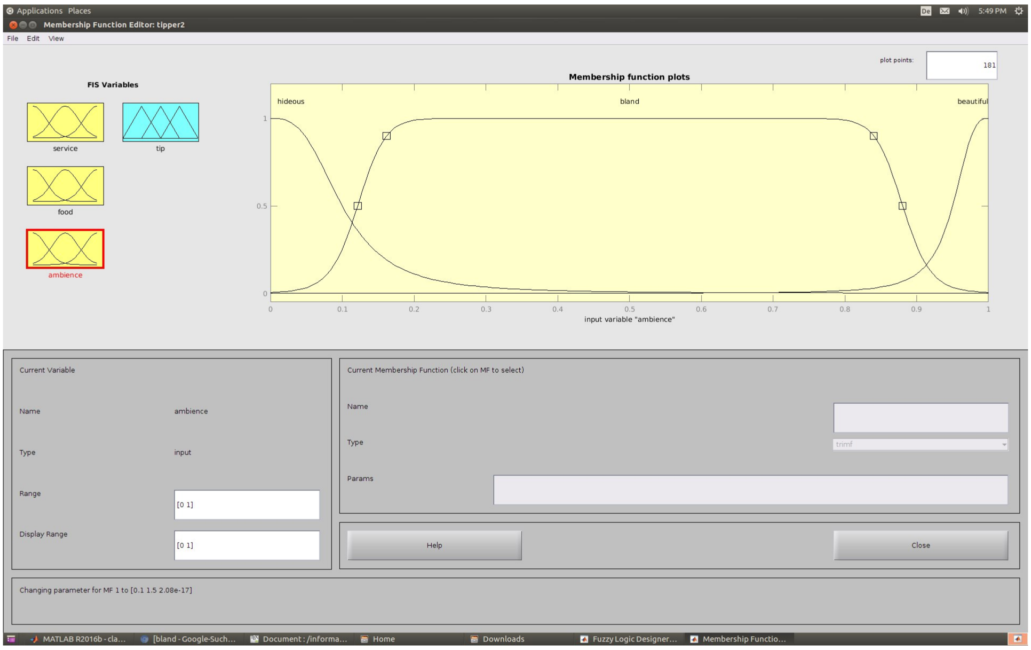
Range: [0 1]

Renaming input variable 3 to "ambience"

Obviously, this new variable has no impact yet, since it has no membership functions and rules associated with it.

1.2 Selection of different membership functions

Membership functions by input variable:
Service has three membership functions; *poor*, *good* and *excellent* which are centered around 0, 5 and 10 respectively. Each of these are gaussian functions which makes sense for such a diverse variable.
Food has two membership functions; *rancid* and *delicious* which hold a plateau from 0 to 1 and from 9 to 10 respectively instead of centering around a maximum value like *service*. This results in two radical sides with a big whole in the middle representing a mediocre taste. This can make sense for food, because lots of people will usually only consider the taste for their foot in their tip, if it was exceptionally good or bad. Service, on the other hand, can have an influence in greater variety.
A sensible way of now including membership functions for our new variable *ambience* could be this:



Name ambience

Type input

Range [0 1]

Display Range [0 1]

Name

Type trimf

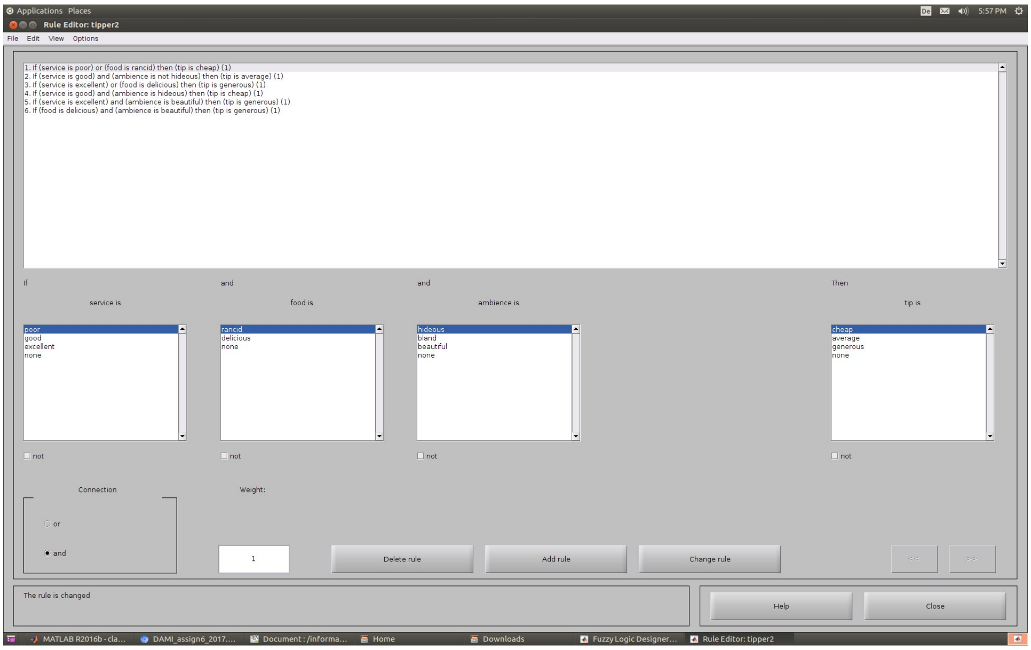
Params

Changing parameter for MF 1 to [0.1 1.5 2.08e-17]

Hideous and *beautiful* are the two extremes, while *bland* serves as the mediocre area in the middle. There is a bit of an overlap between *bland* and the extremes, because these descriptors aren't mutually exclusive. However, *bland* does drop of to zero on both ends, because a truely hideous or beautiful restaurant really has to do something unique to invoke such an extreme impression from a tipper. Lastly, *beautiful* has a steeper drop than *hideous*, because the tipper in my scenario is a bit of a cynic and is more easily disappointed than impressed.

1.3 Creation of rules

The basic rules for food and service are pretty simple: If any of the input variables are in the extremes of their membership functions, then the tip will also be an extreme (unless opposite extremes are met at the same time). On top of that, service has a bit more impact on the final output, because *good* service will result in an *average* tip, while mediocre food has no relation to the tip whatsoever.
Incorporating *ambience* in these rules might look something like this:



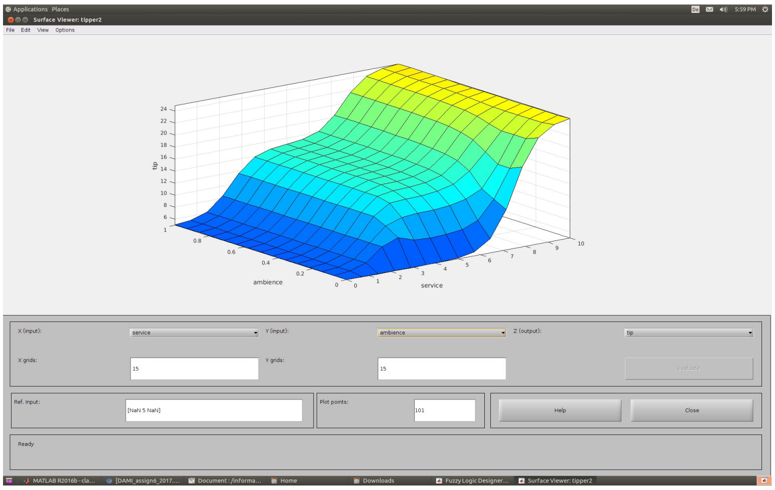
```
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) and (ambience is not hideous) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)
4. If (service is good) and (ambience is hideous) then (tip is cheap) (1)
5. If (service is excellent) and (ambience is beautiful) then (tip is generous) (1)
6. If (food is delicious) and (ambience is beautiful) then (tip is generous) (1)
```

The rule is changed

Here, ambience doesn't have a big influence on its own, but it has the potential to skew the tip in a certain direction, if the corresponding extreme is met.

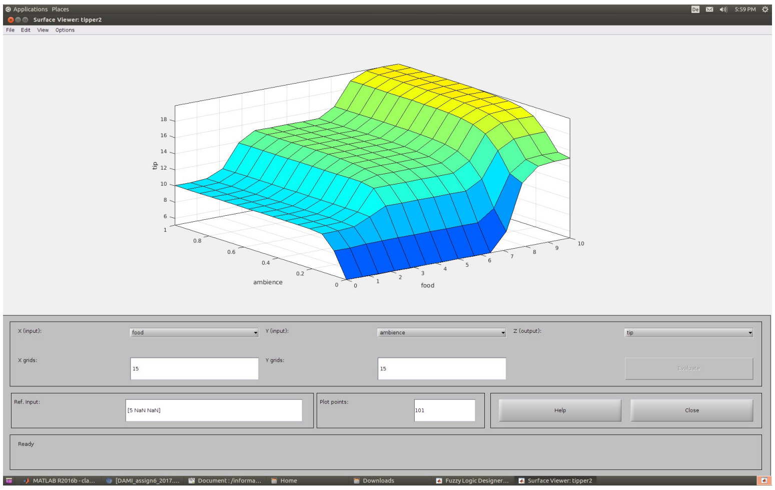
1.4 Visualization

The visualization of *food* and *service* is as expected: The service side has an additional bump in the middle due to *good* service being mentioned in the rules. The food side is flat in the middle due it only having two extremes to consider.
Now to the visualizations of *ambience* with the original two input variables:



Surface Viewer: tipper2

X (input): service
Y (input): ambience
Z (output): tip
X grids: 15
Y grids: 15
Ref. Input: [NaN 5 NaN]
Plot points: 101
Ready



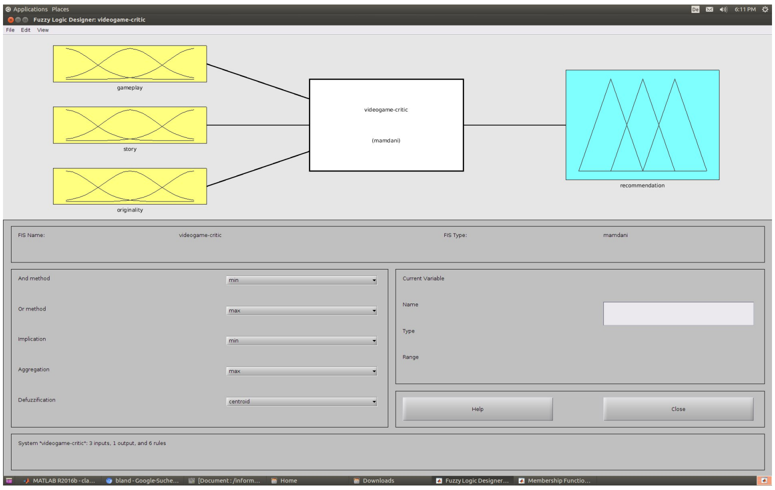
Surface Viewer: tipper2

X (input): food
Y (input): ambience
Z (output): tip
X grids: 15
Y grids: 15
Ref. Input: [5 NaN NaN]
Plot points: 101
Ready

These visualizations are fairly similar except for the edges: On the lower end, great ambience cannot raise the tip, as long as service is at 0, but terrible food can still result in a 10% tip, if the ambience isn't hideous as well.
The same logic applies to the upper end of the graph: Incredible service will result in the maximum tip, even if the ambience is hideous, but amazing food cannot elevate the tip over 15% in that very same ambience.

2

Fuzzy logic model of a videogame critic:
The goal of my model is to determine the audience of a given videogame. A game may be recommended to *nobody*, to *fans of the genre* or to *everybody*. The input variables that influence this recommendation are *gameplay*, *story* and *originality*.



Fuzzy Logic Designer: videogame-critic

FIS Name: videogame-critic

FIS Type: mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

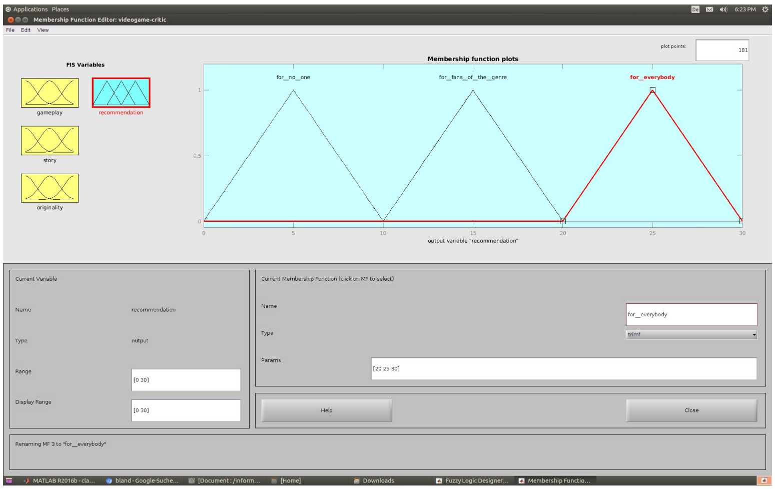
Current Variable

Name

Type

Range

System "videogame-critic": 3 inputs, 1 output, and 6 rules



Membership Function Editor: videogame-critic

plot points: 181

Current Variable

Name: recommendation

Type: output

Range: [0 30]

Display Range: [0 30]

Current Membership Function (click on MF to select)

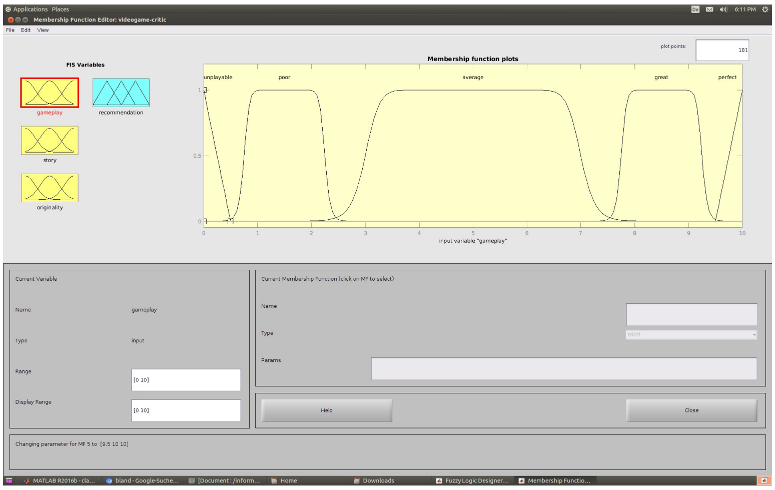
Name: for__everybody

Type: trimf

Params: [20 25 30]

Renaming MF 3 to "for__everybody"

This fuzzy logic system is based on a very specific archetype of videogame critic: Someone who rates gameplay above all else. As long as a game feels good to play from beginning to end, it will probably get a good rating. And since gameplay is so important to this critic, he has a high level of expertise and can judge even the smallest properties of gameplay, resulting in a greater amount of membership functions that a game could land in.



plot points: 181

Name gameplay

Type input

Range [0 10]

Display Range [0 10]

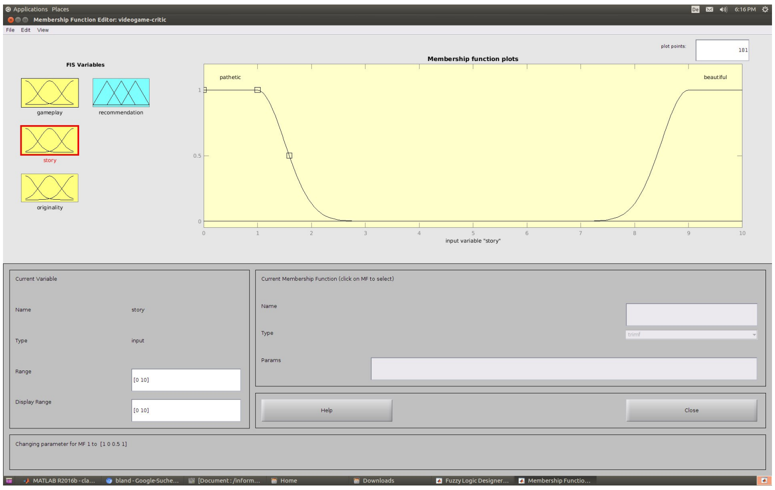
Name

Type trimf

Params

Changing parameter for MF 5 to [9.5 10 10]

But even though this critic focuses on gameplay, he isn't completely oblivious to the other two categories. If a game is incredibly original or has atrocious story, he won't ignore these facts in his rating.
His outlook on stories is kind of boring, with two identical plateaus on both sides.



plot points: 181

Name story

Type input

Range [0 10]

Display Range [0 10]

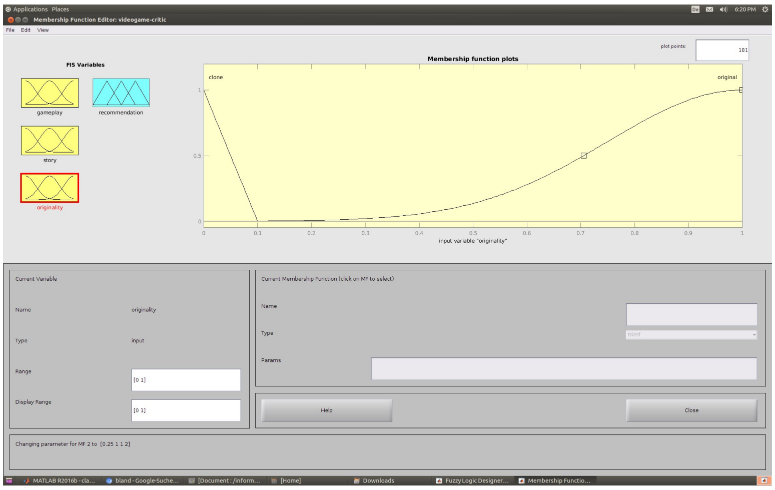
Name

Type trimf

Params

Changing parameter for MF 1 to [1 0 0.5 1]

On the other hand, his view on originality is a bit more unique: If a game completly copied a previous title, it will be very obvious and result in a bad rating. But since videogames are a very iterative medium, he will forgive games that are just very similar to other games, as long as they at least introduce a few new mechanics as well. Naturally, an increased amount of unique mechanics will gradually increase the originality of a title.



Current Variable

Name originality

Type input

Range [0 1]

Display Range [0 1]

Current Membership Function (click on MF to select)

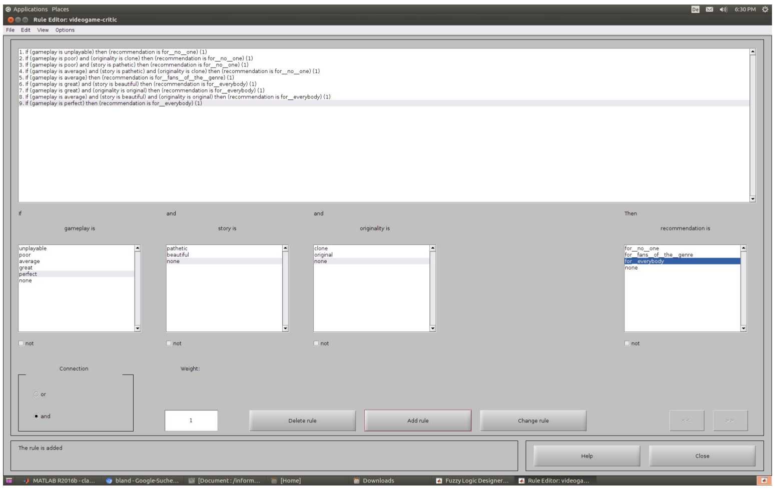
Name

Type trimf

Params

Changing parameter for MF 2 to [0.25 1 1 2]

When it comes to rules, *gameplay* is obviously the driving force among all input variables. An *unplayable* game is not worth most people's time while a game with *perfect* gameplay can probably even be enjoyed by people, who aren't the biggest fans of the given genre.
Story and *originality* do have potential to slighty shift the overall rating on their own. And if both of them happen to be exceptionally extreme in the same direction, they can even counteract an extreme stance of *gameplay*.

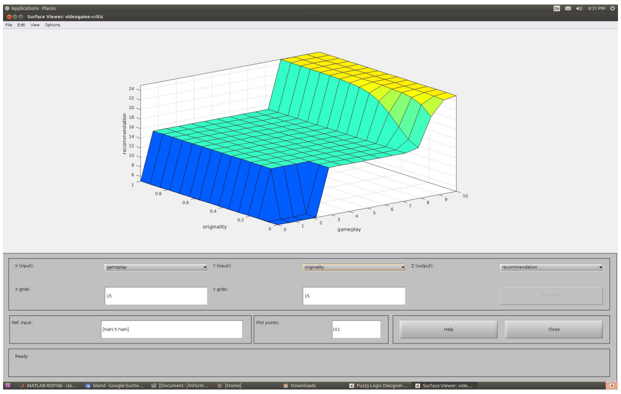


```
1. If (gameplay is unplayable) then (recommendation is for__no__one) (1)
2. If (gameplay is poor) and (originality is clone) then (recommendation is for__no__one) (1)
3. If (gameplay is poor) and (story is pathetic) then (recommendation is for__no__one) (1)
4. If (gameplay is average) and (story is pathetic) and (originality is clone) then (recommendation is for__no__one) (1)
5. If (gameplay is average) then (recommendation is for__fans__of__the__genre) (1)
6. If (gameplay is great) and (story is beautiful) then (recommendation is for__everybody) (1)
7. If (gameplay is great) and (originality is original) then (recommendation is for__everybody) (1)
8. If (gameplay is average) and (story is beautiful) and (originality is original) then (recommendation is for__everybody) (1)
9. If (gameplay is perfect) then (recommendation is for__everybody) (1)
```

Weight: 1

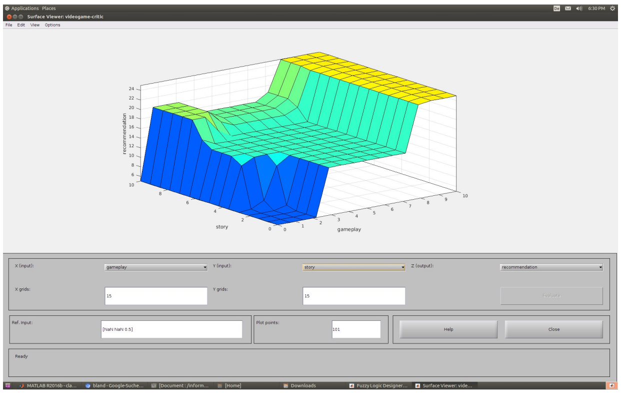
The rule is added

With the membership functions and rules in mind, the resulting visualizations seem to be quite intuitive:



X (input): gameplay
Y (input): originality
Z (output): recommendation
X grids: 15
Y grids: 15
Ref. Input: [NaN 5 NaN]
Plot points: 101
Ready

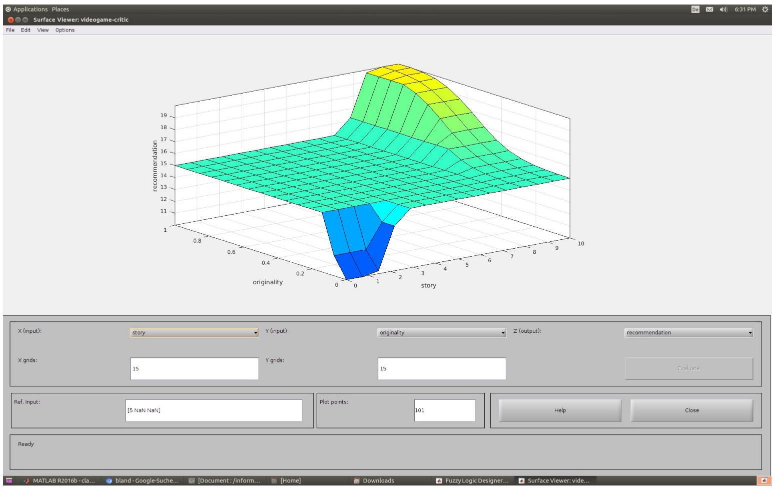
An *unplayable* game stays at 0 no matter how original, but as soon as the game becomes at least *poor*, the originality gets the power to increase the rating to 50%. The huge amount of flat surface on this graph is an indicator of how little the originality actually does for the rating, as long as the game in question isn't in the lower end of originality.



X (input): gameplay
Y (input): story
Z (output): recommendation
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 0.5]
Plot points: 101
Ready

Gameplay and *story* have a more interesting relationship, because *story* doesn't have as much of a steep lower end as originality's *clone* function does. This results in *story* bringing quite a high level of rating without getting past *poor* gameplay. However, gameplay still has the upper hand here and can even bring the rating to a maximum level while keeping *story* at a level of 0.

This surface also holds the only major error of the model: At maximum *story*, the overall rating goes back down around *gameplay* level of 2 before going back up at around 6.5, which makes no sense for any critic and especially not for the one I modeled, because gameplay is supposed to be the driving force here.



X (input): story
Y (input): originality
Z (output): recommendation
X grids: 15
Y grids: 15
Ref. Input: [5 NaN NaN]
Plot points: 101

Finally, we have the relationship of *story* and *originality*. This graph has the most amount of flat surface, which lays on 50% of the maximum rating. This is also quite intuative, since these two variables only have any impact, if both of them are extremes in the same direction, as explained in the rules above. We can also distinctly see the increase that comes from *originality* due to its unique membership functions.