Combining Clustering and FSM models for Adaptive Level Generation

Jorge Diz Pico, David Camacho

***Abstract*—**

I. Introduction

problem: levels are finite
space constraints and design is costly
to solve it: algorithms
that is, pcg
a way to generate levels
rules
is hs been used for all these things in these games
like minecraft
or diablo
even tetris!
adaptive: why?
because it tailors better to tastes
advantages? personalization empoers bonding
hey i could probably find some sweet cites for this
it has been used here and there
sayy, brain training
or, hey, l4d2
our case: the mario competition
what have people used?
passes -> but no link
clusters -> that's cool, but bad random
hopper -> yeah that's cool
so we'll try to do both and cool
have one process guide the other
and an open architecture ripe for flexibility
we'll do this by separating functionality in layers
and encoding levels in transformable ways
urgh this is bad
retake it once the rest is done
you'll have a clearer idea

II. Background

mario is a game about platforms and jumps
and enemies and coins
and you have to get to the end
the mario ai championship is about
plenty of stuff
but the level generation track is about
forming the best level there was
the framework is based on infinite mario by notch
which is
top notch
haha
so yeah
our goal is to make a good architecture for it
that can be flexible, random every time, adaptive, and not too tailored to the game
how did we do it?
keep reading

Jorge Diz and David Camacho are with the Computer Science Department, Escuela Politécnica Superior, Universidad Autónoma de Madrid. c/ Francisco Tomás y Valiente 11, 28049 Madrid, Spain. (email: jorge.diz@estudiante.uam.es and david.camacho@uam.es)

III. Methodology

this is our architecture
there are FOUR LIGHTS
but three layers
two mostly
layer figure
top takes care of identifying the user
how does he like it
second, generating the level
the user identification gudies the process
because
that's the way
u-hu, u-hu
he likes it
u-hu
for the top layer we decided to cluster
it gives a point of reference
he prefers this way, in comparison to others
judging in a vacuum is hard
also makes them feel special
the second, grammars
levels are a succession of repeated basic symbols
also, what comes now relies on what came before
we can model that as a directed graph
an automata, for example
built from the grammar
we can make new "paths", signifying possibilities, parts of the screen
by adding new states or links
automata are flexible
and expressing them as grammars, quite intuitive
they rock
okay then

IV. Clustering

we want to base player profiling based on data
we think it's more faithful
player's don't know what they want
and badly estimate themselves

cite here
or at least steve jobs quote
so let's take data
mario stores two kinds of data
metrics and logs
we'll focus on the metrics
why? because we don't want specific timestamped events
but overarching behavior
so mario challenge stores all this data on the player
they refer to this categories: deaths, blah
we are going to try to infer the different existing styles
so we'll take one sample per person, for these reasons
we need to provide them with comparable challenges
for these reasons
once gathered, we will try to find groups of palayers with similar behaviour
we want three
why three? they are everywhere
experience proves it, but also
the speeder prototype comes from speedruns, light-and-nimble characters, aggresive strategies
explorers come from detail-intensive games, hevy armoured charactered, controllish, slow plans
they form a spectrum, with two extremes
so we decided to model with three, allocating for a mixed intermediate position
also, not a lot of samples
now to choose the algorithm
we decided to rely on the weka library
its cool and awesome and proven. source:
weka offers these algorithms that are the most commonly used.
we'll use this metric to compare them
since it's density blah
we need to blah
we could add a parameter by the user's preferred difficulty
but that's intrusive
and they are bad at it anyway
if you need to choose it before palying, you have no idea
if you've alrady played, we have the metrics thankyou-verymuch
ok so we have the clusters
stored safely somewhere
this process is made so it can be run offline
the clustering of the data, i mean
so then a player comes
and we assign them one of those clusters

V. Grammars

whew that was long
ok so now to fsms
when we assign the player to a cluster, we implicitly assign him a grammar
since each cluster has a grammar associated
the grammars are called schematics and describe how to build the level
in a pseudo-bnf way
take a look at this simple example
HERE BE FIGURE
the derivation rules have weights
a heavier rule has more probabilities of being chosen
the schematics are context-free grammars
and expands on the right
so in the example we start the level
and we can have a pipe or a coin or flat
and we loop
(ok explain this a bit longer)
the schematic is designed to be infinite
to be able to generate levels of any length
this is a sample of a trace
BLA PIPE COIN RICK ROLL
the schematics are done by hand
the terminals are predefined
they are the available "chunks" that the system can build
for example, a small gap
or a pipe
or a couple of blocks
in this case we made every chunk of the same length, two
we wanted the smallest possible to make it more flexible
the arching structure can be obtained with the rules of the schematic
forcing pieces to be together
but if we wanted to control little details, we need little chunks
but we couldn't do one, because pipes for example are two blocks
and gaps of length one are not very mario-like
we needed chunks of the same size so we had a calculable relation
between chunks used and level length
for the number of iterations
we want this length, ok so we iterate this n of times
also, levels of same length, same number of states
so we can compare them
the choice of cluster from the step before
determines what schematic we read
each one has derivations and weights tailored to their needs
but, as we said, same nonterminals
alright so we read the schematic to construct an automaton representing these transitions
we use the parse2 library
we defined a grammar for the schematics
then parse2 reads the schematics and transform the rules into transitions
with their weights of course
the resulting graph
can be seen also as a tree with cycles
see the figure for the tree for our example from before
HERE
in this tree model, we'd be traversing it depth-first
we store the terminals we follow, forming a string of chunks to place
we call this string a "genotype", because it represents the level

two equal genotypes represent the same level
an executor later traverses the chunks
from starting node
choses a random transition
pushes the derivation chain into the stack
then pop top node
if it's terminal, we add it to genotype
if it's nonterminal, we pick a random transition and push it to stack
always proceed with the leftmost element of the chain first
because we build the level from left to right

VI. Results

god damn that was a lot to tell
so what happened
we used social networks, like twitter, facebook and reddit, and word of mouth
to get the data
117 entries
each person was given a package with ready-to-play game
the game would provide a regular random level from the framework
they were asked to just play, with no other worries about completing the stage, or training before hand
and send back the results
we fed into weka
but we filtered some parameters
because the default stage
has no gaps, or enemies other than goombas
so soe parameters were always zero
those parameters:
we used those, though:
then we tried the five clustering algorithms with default settings
except three clusters as explained
(why only the five? er...)
you can see the results of the log likelihood here
em rocked
the three clusters found represented the profiles we hoped
check this sweet table
isn't it gorgeous
plus they are well separated
check this graph
you like it huh?
speeders like this
explorers like that
and my milkshake brings all the boys to the yard
so we made some grammars
here, take a ride in my intermediate
here's the diagram
and a sample trace for good measure
ok a screenshot too
let me explain some design choices
like, this branch here

VII. Conclusions

the separation of layers means that the way you identify users
is not thoroughly linked to how you use that information to guide the building
grammars meant
that we used the same basic chunks
in different ways each
tailoring for styles
grammars are easily intuitive and modifiable
you can control what goes next to what
and how often
and offer choices, different "paths" for the build
we also produced a way of identifying and storing a level
by its genotype of chunks
(did you define chunk? go and do it!)

Future work

we hope
to evaluate this system
with some group control
also
genotypes allow us to use genetic techniques
for example
if the clustering layers defines a player
instad of one membership
with shared membership
we can generate levels from different automata
and mix them
we can also alternate picking derivation rules between grammars by those same percentages

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Innovation under grant TIN2010-19872 (ABANT).