COMPUTER GENERATED GAME CONTENT

OR

YOU MAY BE ABLE TO PAY GAME DESIGNERS MINIMUM WAGE, BUT COMPUTERS ARE EVEN CHEAPER AND DON'T DRINK YOUR MOUNTAIN DEW

Zach Barth
Rensselaer Polytechnic Institute
Introduction to AI, Fall 2007

Modern Game Design

Modern game design calls for the combined efforts of three separate jobs: programmers, who create the game engine and implement it in its technical entirety, artists, who make artistic assets such as 3D models and interface graphics, and designers, who create most of the gameplay content, such as levels, enemies, and items. In an ideal world (I'm being facetious, of course), programmers would be free from the requirement of having artists and designers to help them craft their products; technologies such as procedural graphics and procedural level-generation have helped to provide programmers with their much needed independence, but what about procedural content? A strategy game with procedural content could, for example, feature constantly evolving armies that changed over time, as things changed both in-game and outside of the game. How would one go about accomplishing something like this?

The Problem

Procedural content isn't an entirely new concept – in fact, things like it have been routinely implemented in some genres of games since their earliest days. The rogue-like genre, which began with the text-based mainframe game Rogue, features randomly generated levels and unique items, resulting in a unique adventure every time you play. However, none of this creation was truly "inspired"; instead, the dungeon layouts were very clearly algorithmic, and all items were created in a permutative fashion whereby base item types were combined with pre-programmed effects, all from tables created by game designers. Although this created situations that were technically "unique", nothing it created would ever be surprising. While the designers could have given the algorithms more leeway to "invent" things, there is a high chance that doing so would derail the game from its theme and make such content seem irrelevant. Clearly, a concept of the balance of "signal" to "noise" is important in any situation where content is being randomly generated – but why is this the case?

An Ideal Scenario

Given my survey of books about creativity and AI at the RPI library, most people who know what they're talking about seem to define creativity as the ability to create analogies. A painting, they claim, is simply an analogy of real life (or something to that effect); while I don't entirely agree with this explanation with respect to artwork, I think it does apply to the notion of creating game content – a good deal of games involve taking real life concepts (like the Second World War) and translating them into game systems (in the case of WW2, something done to seemingly no end). There are also a great deal of fantasy games, involving browbeaten memes such as "wizards and

warriors", which could be processed in a similar fashion, or more complex ideas such as zombie pirates, which could theoretically be created by analogizing pirates with zombies and then analogizing the results into game content. We can break this model down to arrive at three required factors for computer driven content creation: the ability to understand human things, the ability to understand game mechanics, and a way to convert between the two. For understanding human things, we'd need some sort of human context database, providing a representation of human ideas in a fashion that a computer could understand. For understanding game mechanics, a system of atomic game operations could lead to interesting results; unfortunately, determining what that would entail is a large task in itself, so instead we will turn to a traditional strategy framework, whereby multiple "units" can exist, move around on a playing field, and interact with each other. To detail the process our algorithm would have to take, we'll briefly work through the RIFLEMAN example.

In the context of the military, we can generalize that a **rifleman** is a type of **soldier**, a human that acts in an **infantry** position and generally wears **body armor**, who carries a **rifle**, a **long range firearm** which must be **aimed**. Hypothetically, we could further parse out the notion of firearms, which shoot bullets, and aiming, which generally requires concentration, and arrive at a game-world analogy, a "unit", that:

- Can move around like other humans (average 1 meter per second).
- Wears armor, which helps prevent damage.
- Carries a rifle, which can shoot bullets at a longer range than other firearms (600 meters), but must be aimed, restricting movement, in order to be fired accurately.

Although this doesn't say much about the algorithm we'd need, it describes the sort of transformations we would like to be able to accomplish. For a computer program to be able to do this on its own, however, would be incredibly non-trivial – not only would it need to fully understand the human context going in, but it would also have to fully understand the products of combined game rules and values and be able to compare and rank the accuracy of the input to the output. Yikes.

Since a project of that scope is far beyond anything I could accomplish in a few weeks at the end of the semester, I took a simpler approach to implementation and ran with it, seeing what sort of content an agent that skimmed Wikipedia could create.

The Algorithm

As we previously established, a system to generate content requires three components: a human context database, a system of base mechanics, and an algorithm to translate HCD entries into pieces of game content. The program designed for the task, RICHARD-GARRIOT, used Wikipedia, an online encyclopedia, and WordNet, the Princeton semantic network, in conjunction to act as its HCB. Game mechanics were described through grammars, which shaped the output of the HCB processing in a way that they could be easily incorporated into a game, which in our case was a small, cliché dungeon-crawler called *Wikipedia Quest* that I created solely for this purpose.

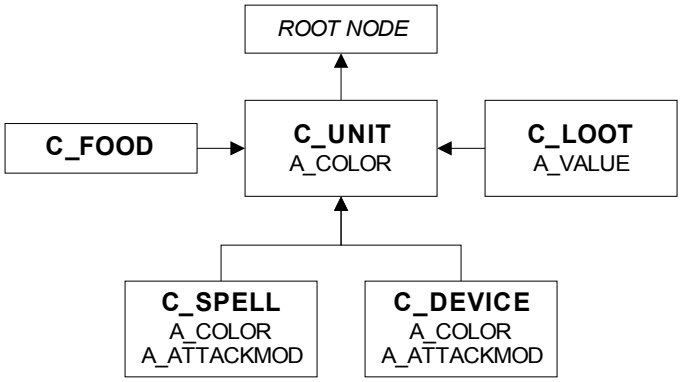
Using Wikipedia as the main source of content has some advantages, namely that I don't know of anything else that would work better, but also presents many challenges. The existence of links between pages allows for some cross-topic connections to be established, but they typically tend to exist only for readers, and tend not to give hints about the meanings of the content on the page. Unfortunately, the same can be said about *all* the text on the page – it was designed for readers, and not to be parsed by a machine trying to learn about the idea a word describes. To fully understand what a page says, we'd have to use some sort of natural language parsing, another topic that falls far outside of the scope of this project and is still in the domain of things that haven't fully been figured out. Although the Python Natural Language Toolkit (NLTK) appears to have some functionality that could have helped, it seemed too complicated to work into the project on my limited time frame, reinforcing my decision to choose something else.

Statistical analysis, an alternative to lexical parsing, proved to be far simpler than parsing; within the domain of Wikipedia, I was able to make a few assumptions that helped to guess the relevancy of specific words. First, I assumed that most statements were in the positive sense; thus, if you look up Zebras on Wikipedia, you are more likely to find the phrase "zebras are black and white" than the phrase "zebras are not pink", which allowed me to freely assume that only relevant and "true" words appear on a page. Additionally, I assumed that important descriptive words were located closer to the top of the page than at the bottom, as most Wikipedia entries seem to conform to this convention. The final statistical assumption was that uncommon words have more meaning, something that seems to fit with my primitive knowledge of information theory. Unfortunately, I was unable to find a decent dataset of word frequencies, but did apply this assumption in the form of a database of "stop words" which were filtered out before any processing.

As the core algorithm of RICHARD-GARRIOT was designed to be independent of the game I would use the content in, it would have to accept a description of possible outputs that were

compatible with the target system. This was accomplished through the notion of "game grammars", which encompasses the possible structures of content pieces and mechanics that can be applied to them. While the system could hypothetically generate mechanics through a system of atomic game operations, I chose to design the core mechanics, such as the ability of a weapon to inflict fire damage that could set a unit on fire and cause it to continue to receive burn damage, and have RICHARD-GARRIOT apply them where relevant, such as to a flamethrower or a candle.

Game grammars are structured as trees, with each node corresponding to a topic (page) on Wikipedia. Starting by matching a topic to the root node, a domain tree is generated, hopefully describing our root topic in a way that matches the game grammar. A child of a node can be any word or topic link that appears on that node's page, which can then be looked up itself on Wikipedia, resulting in a recursive algorithm that creates the fullest domain tree possible for the given grammar and root topic. Since we don't want just anything to appear as a piece of content or sub-content, I added the ability to add qualifiers to nodes – as we're working with words and phrases, this is where WordNet comes into play. Thus, each node may specify a class type, which is defined as a list of hypernyms (a set to which the word belongs), at least one of which the word must belong to in order to classify as belonging to the class and thus make it into the resulting domain tree. Going back to the RIFLEMAN example, we could specify a weapon class, to which anything with WordNet entry 04565375N ("*weapon, arm, weapon system; any instrument or instrumentality used in fighting or hunting*") as a hypernym would belong. Entry 04090263N, "rifle", belongs to this class, through the hypernym chain "rifle" -> "gun" -> "firearm" -> "weapon", but entry 07695965N, "sandwich", would not.



Class Name	Class Hypernyms
C_LOOT	Crystals, Currency, Treasure
C_FOOD	Food
C_UNIT	Living Thing, Spiritual Thing, Vehicle
C_SPELL	Discharge, Flash
C_DEVICE	Device, Weapon, Natural Object

Game Grammar for Wikipedia Quest

When scanning a page for child nodes, we sometimes will pick up a word or phrase and desire to use it as a content node; when processing the page “Greek Gods”, we would like the word “Zeus” to register as a child that qualifies under the C_UNIT class, as “Zeus” is pure content. Some other words, though, such as Zeus being **male** or a flamethrower shooting **fire** hint at things that represent game mechanics we wish to implement for a piece of content instead of content itself. This is achieved in RICHARD-GARRIOT through the use of attributes, which are defined as a set of values and a list of hints for each value, and are applied to any nodes we wish to apply the mechanic to if it is considered relevant.

Attribute Name	Attribute Value	Hint Words
A_VALUE	VALUABLE	collectible, expensive, heirloom, important, invaluable, precious, prized, scarce, treasure, treasured, valued
	WORTHLESS	worthless, cheap, inferior, insignificant, mediocre, paltry, trifling, trivial, unimportant, useless, valueless, waste
A_COLOR	RED	red, burgundy, crimson, fuchsia, magenta, pink, puce, scarlet, terra cotta, titian, vermeil, vermilion
	BLUE	blue, azure, beryl, cerulean, indigo, teal, turquoise, ultramarine, aquamarine
	YELLOW	yellow, amber, gold, golden, ochroid
	GREEN	green, beryl, chartreuse, jade, lime, malachite, moss, olive, verdigris, vert, viridian
	PURPLE	purple, violet, amethyst, bluish-red, heliotrope, lavender, lilac, magenta, mauve, mulberry, orchid, perse, plum, reddish-blue, violaceous
	ORANGE	orange, peach, red-yellow, titian, tangerine
	WHITE	white, caucasian, achromatic, achromic, alabaster, blanched, bleached, frosted, ghastly, hoary, immaculate, ivory, light, milky, pallid, pasty, pearl, silver, silvery, snowy
	BLACK	black, dark, dusky, ebony, coal, sooty, slate, raven, onyx, obsidian
	GREY	grey, gray, ash, ashen, cinereal, clouded, dingy, dove, drab, dusky, dusty, granite, heather, iron, lead, leaden, livid, mousy, neutral, oyster, pearly, peppery, powder, sere, shaded, silvered, silvery, slate, smoky, somber, stone
	BROWN	brown, amber, auburn, beige, bronze, sienna, khaki, henna, puce, terra-cotta, tan, rust, umber
A_ATTACKMOD	BURNING	burn, fire, blaze, incinerate, kindle, hot
	ACIDIC	acid, acidic, acrid, caustic, alkaline, corrosive
	FREEZING	ice, cold, chill, freeze, chill, frost
	SWIFT	fast, quick, agile, elude, evade, swift, dodge
	BLUDGEON	bludgeon, bash, beat, clobber, whack, strike, pound, pommel, batter
	PIERCING	gash, incision, jab, pierce, piercing, prick, puncture, thrust, incise, stab
	SLASHING	carve, chop, gash, hack, injure, lacerate, rend, rip, sever, slice
	HEALING	alleviate, ameliorate, doctor, medicate, reanimate, rebuild, recover, regenerate, rehabilitate, rejuvenate, remedy, renew, renovate, repair, restore, resuscitate, revive, revivify, salve, soothe

Attribute Listings for Wikipedia Quest

Although I would have liked to use a thesaurus to automate the discovery and application of attributes, I was unable to find a thesaurus file that had any sort of relevance indicator. According to the thesaurus data-file I did find, the word “blue” was marked as a synonym for “red”, something

that is clearly unacceptable for our purposes, meaning that, at least for this iteration, all synonyms would have to be entered by hand.

The end result of this process should hopefully be a domain tree full of content, ready to be spliced apart and inserted into our game. Key node points, which for our example were the instances of the C_FOOD, C_UNIT, and C_LOOT classes, provide cutting points, at which we can detach them from the tree and feed them into product lists. A large Wikipedia entry, however, could contain hundreds of products, many of which we may consider inferior; if it were possible to guess at the relevance of a node, we could pass that information into the products and cull out products that are irrelevant (such as mistaking the word wrench, when used in the context of pulling a muscle, for the hand tool, which would qualify as a C_DEVICE) or boring (such as a unit that has no attributes or actions).

The rough heuristic used by RICHARD-GARRIOT to create the content for *Wikipedia Quest* was to multiply a word's percentage distance from the bottom of its parent page by a percentage scaling of the word's WordNet sense number when used in the context of the class for which it matched. For example, consider having reached the topic "man" when qualifying a C_UNIT node. While scanning the page, we come across "spear", something that registers on WordNet as being a C_DEVICE, due to having entry 04565375N, "weapon", as an inherited hypernym. "Spear" occurs as the 375th word on the "man" page, which contains a total of 2904 words, and when used as a weapon is the first (and therefore more frequently used) sense of the two senses of the word "spear". This would result in a relevance score of ((2904-375) / 2904) * (2/2), or 0.871, a relatively high relevance. If the word occurred lower on the page, the relevance would be driven down, as would our definition of "spear" being less common than the alternative definition of the word, "n implement with a shaft and barbed point used for catching fish", which is only classified as an "implement". When ranking a product with sub-nodes, the child node scores would be added onto their parent's, with any attributes adding slight bonuses in order to reward complexity. The highest ranking, unique products were selected to be in the game.

Example Output

To demonstrate the output capacities of RICHARD-GARRIOT, the following are the (annotated) results of feeding the topic “Punic Wars” in as the root topic, creating a “Punic Wars” domain tree. In reality, the domain tree was far larger than is present here, but while it may make my paper seem larger, 20 pages of computer printouts accomplishes very little.

```
'Punic Wars' DOMAIN TREE
-- 'Diocletian' 0.89 P_UNIT C_UNIT
-- -- ('A_COLOR', 'A_COLOR_WHITE', 'silver', 0.34756097560975607)
-- -- 'Crisis of the Third Century' 0.37 null C_DEVICE
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_BLUDGEON', 'beat', 0.7901785714285714)
-- -- -- ('A_COLOR', 'A_COLOR_WHITE', 'silver', 0.49107142857142855)
-- -- 'Empire' 0.19 null C_DEVICE
-- -- -- ('A_COLOR', 'A_COLOR_GREY', 'lead', 0.73057733428367788)
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_SLASHING', 'carve', 0.25659301496792586)
-- -- 'lightning' 1.77 null C_SPELL
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_BURNING', 'hot', 0.25714285714285712)
-- -- 'bolt' 1.77 null C_SPELL
-- -- 'Jupiter (mythology)' 0.63 null C_DEVICE
-- -- -- ('A_COLOR', 'A_COLOR_GREY', 'lead', 0.74210526315789471)
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_HEALING', 'rebuild', 0.35087719298245612)
-- -- 'Hercules' 0.32 null C_DEVICE
-- -- -- ('A_COLOR', 'A_COLOR_WHITE', 'silver', 0.99763593380614657)
-- 'Roman Navy' 0.32 P_UNIT C_UNIT
-- -- ('A_COLOR', 'A_COLOR_GREY', 'lead', 0.44516477579686659)
-- -- 'First Punic War' 0.31 null C_DEVICE
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_SWIFT', 'quick', 0.68802781917536016)
-- -- -- ('A_COLOR', 'A_COLOR_GREY', 'lead', 0.3979135618479881)
-- -- 'counter' 0.74 null C_DEVICE
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_SWIFT', 'fast', 0.74891774891774887)
-- -- 'Corvus (weapon)' 0.96 null C_DEVICE
-- -- -- ('A_COLOR', 'A_COLOR_BLACK', 'raven', 0.99111111111111116)
-- -- -- ('A_ATTACKMOD', 'A_ATTACKMOD_PIERCING', 'pierce', 0.78666666666666663)
-- -- 'spike' 1.92 null C_SPELL
-- -- 'bridge' 0.88 null C_DEVICE
-- -- -- ('A_COLOR', 'A_COLOR_GREY', 'lead', 0.93916913946587532)
-- -- 'ram' 0.95 null C_DEVICE
-- 'fifty' 0.05 P_LOOT C_LOOT
-- -- ('A_COLOR', 'A_COLOR_YELLOW', 'gold', 0.37992831541218636)
-- 'board' 0.54 P_FOOD C_FOOD
...
```

After creating a domain tree, RICHARD-GARRIOT then prunes at the key nodes (C_UNIT, C_FOOD, and C_LOOT) to create our content pieces, which can be seen below. Thanks to the somewhat helpful heuristics, the following units were automatically picked out as some of its best products. While a “board” may not be very tasty and “lead” continues to comprise what seems like half the game’s content, we can imagine Diocletian shooting lightning bolts and Carthaginians stabbing people with their bronze swords. With an algorithm this primitive being somewhat successful, perhaps the notion of computer created content isn’t that unrealistic?

'Punic Wars' CONTENT PIECE LISTING

```
'fifty', 'C_LOOT, 0.05
      'A_COLOR', 'YELLOW', 'gold', 0.38

'board', 'C_FOOD', 0.54

'Diocletian', 'C_UNIT', 0.89,
      'A_COLOR', 'WHITE', 'silver', 0.35
      'lightning', 'C_SPELL', 1.77
            'A_ATTACKMOD', 'BURNING', 'hot', 0.26

'Carthaginian', 'C_UNIT', 0.49,
      'A_COLOR', YELLOW', 'gold', 0.90
      'sword', 'C_DEVICE', 0.87
            'A_COLOR', `BROWN', 'bronze', 0.95
            'A_ATTACKMOD', 'PIERCING', 'thrust', 0.94

'Roman Navy', 'C_UNIT', 0.32
      'A_COLOR', `GREY', 'lead', 0.45
      'spike', 'C_DEVICE', 1.92

'Philip V of Macedon', 'C_UNIT', 0.33
      'A_COLOR', 'GREY', 'lead', 0.24
      'lead', 'C_DEVICE', 0.53
            'A_COLOR', GREY', 'lead', 0.98
            'A_ATTACKMOD', 'ACIDIC', 'corrosive', 0.80

...
```

The End Product

As this project wouldn't be complete until I was able to actually use RICHARD-GARRIOT to generate content for a real game, I designed a cliché role-playing game, *Wikipedia Quest*, in the style of a retro dungeon crawler where every room was based on random topic from Wikipedia. Each room would contain enemies for the player to fight, food to be picked up and eaten to regain energy, and loot to increase the player's score; all of these were taken directly from the domain tree for the room's topic generated and pruned using the algorithms described above.

To make the game more interesting, I programmed RICHARD-GARRIOT to also pull down an image from Google Image Search by searching for the topic that the content was based off of. Thus, when picking up "lunch" in the "Microsoft Interview" room, the player is shown a sandwich sitting on a plate, which was the first image to come up on the GIS search page for "lunch". While this process was fairly trivial, it introduced a few interesting difficulties, with the most significant being a need to manually sort through all of the images gathered and delete those that were pornographic (quite a few).

One thing that I discovered by playing through the game with the generated content was that my relevance algorithm did little to stop homographs (different words that are spelled the same) where the incorrect match was a frequently used sense. The worst offender was the word "lead", which has many completely different meanings, but yet frequently occurred registered as a high scoring C_DEVICE, from what I believe was a combination of a medium frequency and being "grey" and resistant to "corrosion" (a hint word for ACIDIC). While this could be solved by using non-ambiguous words or a language that lacks homophones, I imagine that using proper language processing would be equally effective, as we could hint that the usage of "lead" in "Jason lead the Argonauts to victory" was as a verb and thus should be discarded, even though "lead" as a noun may be a high ranking C_DEVICE we would like to use.

While the results ended being somewhat schizophrenic, I urge the reader to give *Wikipedia Quest* a try to fully comprehend what it is like to play a game with entirely computer generated content. An installer can be downloaded from the following location:

http://zachtronics.emala.net/files/WikipediA_Quest_Installer_1.2.exe