GAMASUTRA
The Art & Business of Making Games

PRINT

The Art of Rogue Legacy, or "why less is more"

By Glauber Kotaki on 08/12/13 01:56:00 pm

You may have recently heard of the game, *Rogue Legacy*. It's a game about a cursed family traversing a vast and ever changing castle throughout generations. It has randomly generated areas, a constantly growing manor, a variety of differing enemies and a whole lineage of characters to choose from. For a single contracted artist, this might sound like a nightmare because of so many things going on, but it was actually quite the opposite.

Working with *Cellar Door Games* developers, Kenny Lee and Teddy Lee was a pleasant surprise: after playing the game, I never expected it to be so big - 15-hours+ for some people just to beat the guy – all with so few assets I delivered over one year. Being a roguelike with procedurally generated levels sure helps, but with fresh and interesting design as its pivot, art could easily be minimized to help a small company make a great game or even help it from going bankrupt.

Retro is cheap (in all meanings)

A lot of games today go for resized pixel art for a lot of reasons. There's a recurrent passion for 'retro' from players, and with high definition resolutions and advanced technical resources today, giving a modern visual twist on them is just a matter of taste. It's noticeable on great games like *Superbrothers: Sword & Sworcery EP*which mixes double and triple sized pixel art with a full resolution UI or vector-like special effects; or *FEZ*, which adds simple 3D polygons that mimic an 8-bit style to the scene. Of course there are also games that remain loyal to the pixel, like *Spelunky* (the first release) or *Oniken*. And technically, all of them can make the game smaller, having spritesheets that can be resized through code - and that's a good start.

Double or even triple sizing graphics stress that retro feeling, with strategically placed pixels, at very low costs. Generally speaking, pixel art is pretty cheap compared to most artistic techniques. Naturally it's possible to find art in a pixel game that costs more than some lower-poly 3D games, but overall it is one of the more accessible art resources around - no wonder almost all people who have contacted me for pixel art are indie developers.

Cellar Door Games was one of them, having reached me because they wanted to make a 2D *Demon's Souls*, using pixel art as a safe but effective way to make the game look good (or at least I hope people think my art looks good).

For *Rogue Legacy* specifically, most of the art in-game was double-sized. Some character traits like dwarfism or gigantism scale the character at different values, as do some enemies depending on their difficulty tier. Environment

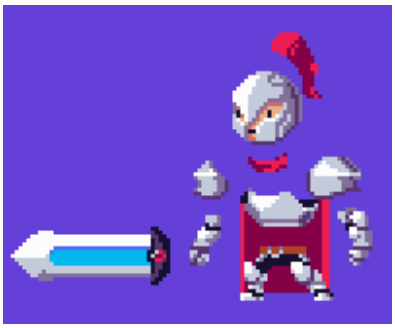
tiles are also double-sized, but it was not always like this (I'll get into that). Interface, on the other hand, was primarily drawn in full resolution. It was difficult to find medieval fonts that matched well enough with pixel art - and I didn't want to get the obvious bitmap font choice - or a nice, readable one. Having a mix of full resolution and pixel art was OK for us, because we were not really into going for a loyal 'retro' anyway - there's no size limit on animations or tiles, or colour limit on palettes - so that was OK.

I also believed there was a chance the UI could change greatly once we received feedback (as I have experienced on previous projects), so it would be handy to have vector based sources and 'smart objects'. It actually didn't change that much in the end, but having bigger UI assets allowed us flexibility in getting different sized but simple shaped dialogue boxes and pop-ups - this also facilitates a homogeneous result for art direction.

Going by parts

Even though the UI stayed mostly static, the funny part is what did change a lot during development was the knight design. Getting a whole family tree of characters in the game was the most complicated visual issue we had. With each new child, we wanted to give players the feeling they were playing with a completely different person. Design-wise there was an elegant solution, with randomized classes, spells, and traits. Visually though, it was more of a challenge.

Luckily enough, when we first started discussing the knight's design we made the assumption he would have different looking gear. I say luckily because *Rogue Legacy*'s development was very iterative, and we had no idea it would be a "roguelite", or have different characters, or even classes. Either way, his initial design was broken in body parts.



As shown in the image above, each armour part is a different group of layers, with each layer containing a different frame in the animation (for example, a "sword" folder with several layers showing different angled swords). This way we could get more animation just by changing position rather than drawing more frames. As our priority was time and cost, this was extremely useful as revising animations was just a matter of pushing layers around. In fact the whole technique turned out so well that I now use it on current projects with even more fluid animations.

Having a separated body for the main character, Kenny could treat each part as a different entity. This way we could get different armour sets just by partially tinting each gear accordingly, all with one armour design. The process also allowed us to make more designs, going for five more armour sets, randomizing helmets, chest plates and shoulder pads (gauntlets, greaves and the cape were too small or subtle to have any noticeable change).

Also, many of the animation sets barely changed between animations, which was very handy as only one or two frames were needed for each new part.

With such a flexible layer system for both art and code, we went even further making small details for character classes (such as the Barbarians' horns, Mages' beard or Paladins' shield). Some even use the tinting feature, such as the shadow-faced Assassins and pale Lich Kings. We originally planned different weapons for classes, but that was not really a good solution by itself because there was no obvious weapons for all classes (what would a Miner use? How could we tell an Assassin and a Paladin apart?). We might have been able to add weapons along with the small details, but time and money was running low at that point, so we had to choose one or the other.

You can check all features applied below: armour parts differently tinted, skin tinting, class assets and gender swap (a later community suggestion):

Both tinting and layer features were used on enemies too, at a smaller scale, to tint specific parts or just the whole sprite to show stronger versions of them.

Keep it homogeneous

At first, *Cellar Door Games* was going to hire another artist to make the environments. It did not work out as well as firstly figured, as it took too much time just to find the right person and get assets done at the same pace as characters. Not only that, early feedback told us that the environments were not meshing well with the rest of the game because it was not pixel art, but illustrations using full resolution - unlike UI, which could be abstracted since it was not in direct context with the actual game.

As I had been previously tasked at making in-game scenery props like tables, chairs and bookcases, I decided to give full backgrounds a try. Because it came from the same artist and it was pixel art, we no longer had that problem of environments not matching the knight and enemies.

Needless to say, we used tilesets and discernable palettes to compound each area. They basically consist of 3 tiles: background, foreground and floor, so we could optimize production, implement them as fast as we could, and thumbs up the results.

Those scenery props are actually all the same entity, just using different sprites and collision boxes - all four areas have its own tables, for example. This way we did not have to manually place different props for each area, which would have taken much more time to do.

Scrapped

Things that never saw the light of day: combo system! Yep, instead of that single swing we actually had a three-hit combo. The one we kept was actually the third, heavy hit. Having a more complex combat system didn't fit well with the rest of the gameplay, and balancing it wasn't easy. In the end we stuck with a good-ranged attack as the

standard attack animation.

Some enemies were nerfed during development. Ninjas were supposed to always turn into logs when hit, appearing above you in a down thrust attack. The only way to kill them was to dodge this attack and counter-attack. And you think those floating wizards are cheap? Well, originally they were supposed to teleport.

Lastly, like a lot of games, a number of enemies never made it into the final version of *Rogue Legacy*. Some enemies were scrapped because they did not fit well with other enemies, since their behavior is heavily based on how they could be placed with other foes in a room, and others were just too buggy.

A simple yet graceful bat flying around (replaced by flaming ghosts and haunting skulls because of their versatile movement). A spider-like monster hand and an energy spark thingie straight from Dinosaur Island (not really) which crawled onto walls and ceiling. Undead turrets (they still looked like they could be destroyed, so we chose simple metal cannons) and what I like to call the 'Energon', a counter-attack based enemy so complex it could even be a boss.

Conclusions?

Rogue Legacy is all about design as it aims to be, before anything else, a fun game. Characters don't really need ultra-smooth animations to run like brave idiots and die like drama queens. The menacing castle just need a garden, a tower and a dungeon to be deep and rich. Less may be more, but quality sure comes first than quantity.

I hope this experience with *Rogue Legacy* motivates small developers, too. Given the success the game has gotten, and being just a few spritesheets, numbers and sounds from a five-member team, *Rogue Legacy*proves a game doesn't need to be extremely beautiful or a lot of resources to be good.

Thanks goes to Kenny Lee for helping me get all development details and grammar correct!

Return to the full version of this blog