Data-Driven Map Generation

Marrying Database Systems and Video Games Engines

We demonstrate the similarities between problems handled in video game engines and database systems. Video games deal with rapid computation over vast amounts of data in many areas. We claim that tying in database systems can improve the performance of those areas. To exemplify this, we demonstrate data-driven map generation.

Maps Can Not Be Random Spaces

Maps have to cater to the type of game and can not be fully random to be enjoyable. But for each game, building blocks can be conceived of which a map can be comprised.

Part of a dungeon from Nintendo’s game The Legend of Zelda: A Link to the Past. The design is room based, where rooms are connected to each other through doors. Rooms are enclosed spaces with functional and decorative elements. Either whole rooms could form building blocks to create random dungeons, or smaller elements, like the pathways on the lower right room form building blocks to generate random rooms.

Maps in Video Games

Growing the Map

The modules in the following example are composed of the tiles water ( ), coast ( ), walkable ( ), and walls ( ).

Core Algorithm for an Outward Expanding Map

1. From a set of available modules \( M \) take one module as seed \( S \). The seed can either be randomly selected or passed as parameter. (This step is not pictured above.)
2. Select the outermost blocks of the map generated up to this point, using the user defined function banlieues().
3. Select the outer edges \( E_{\text{exp}} \) from those blocks, together with the direction they are facing. Again, we are abstracting this process into a UDF edge().
4. Select the outer edges \( E_{\text{local}} \) from all modules in \( M \), together with the direction they are facing. We can use edge() for this, too.
5. Find edges in \( E_{\text{exp}} \) and \( E_{\text{local}} \), such that they face opposite directions and can be joined on the table of compatible tiles \( C \). If one edge in \( E_{\text{exp}} \) can be paired with multiple edges in \( E_{\text{local}} \), pick one at random. The column freq controls the probability of a matching row in \( C \) to be used as glue in situations where we can choose between multiple join partners.
6. Repeat steps 1 through 5 until a termination condition has been reached, the default being having reached a certain map size.

Why Stop Here?

More (All?) Parts of Game Engines Yarning for SQL

Not only map generation can benefit from borrowing from SQL! In future work we will move more components of the game engine over to the database system. More candidates that are typical components of a game engine are:

1. Incremental simulation of physics, i.e. applying trajectory vectors to objects in bulk,
2. Collision detection,
3. Pathfinding,
4. Control of non-player characters (NPCs or “AI”),
5. Determining visual objects during rendering (culling),
6. …