Written report: UML and Procedural Content Generation

Homework by

Bennet Kuhlmann

Game Design: Games Programming

Semester IV

Lecturer: Kevin Hagen

University of Applied Sciences (UE)

Hamburg

28.06.20

UML

Task 1:

UML stands for Unified Modeling Language and describes a modeling language in software development. It is mainly used to visualize, specify and plan the development of software with the added benefit that one can visualize how the software is supposed to work. This makes it easier to figure out where to start development and ensures that the development process won't descent into utter madness and chaos.

Task 2:

Relationships

Realization:

This relationship describes the act of the client using a method of the supplier.

Composition:

Composition describes the relationship between the container and its contained elements.

Inheritence:

Inheritence describes the relationship between two classes/elements that inherit certain features from each other.

Association:

The Association describes a dependent and bonded relationship between two elements.

Dependency:

Dependency describes the relationship between two objects/classes/elements. If one of the elements changes, the other element will change as well.

Procedural Content Generation (Roguelike/-lite)

Task 1:

Procedural Content Generation describes the creation of pseudo-randomized content generated by a program based on previously set parameters. This may include level generation, item generation, NPC generation (etc.). This kind of content generation is mainly found in the rogue-like/-lite genre, where it basically functions as the core of these kind of games, as they heavily rely on procedurally generated worlds and items. Notable examples include: The Binding of Isaac, Slay the Spire, (Cosmic Dissonance (shameless self-advert)), Risk of Rain and more.

Task 2:

Well, true randomness is hard to decently generate, causes a load of problems gameplay-wise and worsens the feeling of playing a game. For example, generating a level and having one of the tiles the program is supposed to generate being a wall because “huh uh random”, leads to a possibility of being unable to actually PLAY the game or progress whatsoever.

Pseudo-randomness on the other hand is based on algorithms and parameters that were previously set, for example, the level generation process pick tiles out of a determined pool

of tiles, which does not include a wall tile. Another example would be item pools, out of which the program picks one. Some games create seeds, a code which can be entered to force a previously determined sequence of actions to happen in the game itself.

Task 3:

UnityEngine.Random is simpler, easier to understand and easy to implement. System.Random is komplex, harder to implement and may cause issues when used by someone who doesn't know how to work with it. However, System.Random has more options in what you can actually do with it.